UNIVERSIDADE FEDERAL DO PAMPA

EDUARDO CARVALHO TEIXEIRA

PROJETO DE UM MÓDULO CLIMÁTICO PARA UM SIMULADOR DE GESTÃO RURAL

Bagé
2019

EDUARDO CARVALHO TEIXEIRA

PROJETO DE UM MÓDULO CLIMÁTICO PARA UM SIMULADOR DE GESTÃO RURAL

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Engenharia de Computação como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientadora: Ana Paula Lüdtke Ferreira

Bagé
2019

Ficha catalográfica elaborada automaticamente com os dados fornecidos pelo(a) autor(a) através do Módulo de Biblioteca do Sistema GURI (Gestão Unificada de Recursos Institucionais).

266 Teixeira, Eduardo Carvalho

Projeto de um módulo climático para um simulador de gestão rural / Eduardo Carvalho Teixeira.

129 p.

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Pampa, ENGENHARIA DE COMPUTAÇÃO, 2019.

"Orientação: Ana Paula Lüdtke Ferreira".

1. Simulação. 2. Modelagem. 3. Meteorologia. 4. Climatologia. 5. R. 6. Unity. I. Título.

EDUARDO CARVALHO TEIXEIRA

PROJETO DE UM MÓDULO CLIMÁTICO PARA UM SIMULADOR DE GESTÃO RURAL

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Engenharia de Computação como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Trabalho de Conclusão de Curso defendido e aprovado em: 26 de novembro de 2019.

Banca examinadora:

Profª. Drª. Ana Paula Lüdtke Ferreira
Orientadora

Prof. Dr. Bruno Silveira Neves
UNIPAMPA

Profª. Drª. Sandra Dutra Piovesan
UNIPAMPA

Dedico este trabalho aos meus pais e irmã.

AGRADECIMENTO

Agradeço aos meus pais, Edina Vasconcelos de Carvalho e Laercio Vasconcelos Teixeira por sempre estarem ao meu lado, fornecendo todo apoio e carinho necessário.

Agradeço também aos amigos que fiz durante o período em que estive morando em Bagé, em especial aos colegas: Ânderson Fischoeder Soares, Daniel Meirelles Affeldt, Gabriel Ladeia Costa e Ricardo Peixoto Robaina.

Por fim, agradeço aos professores da UNIPAMPA que contribuíram para a minha formação até o momento, em especial a minha orientadora Ana Paula Lüdtke Ferreira e demais professores da Engenharia de Computação.

RESUMO

Este trabalho apresenta o projeto de um módulo climático para um simulador de gestão de propriedades rurais no formato de jogo sério, construído com uma arquitetura modular que faz uso do paradigma de orientação a objetos sobre o motor (engine) de desenvolvimento de jogos Unity. Aspectos meteorológicos e climatológicos influenciam diretamente no crescimento da vegetação e comportamento dos animais. A gestão de uma propriedade rural está, assim, diretamente ligada às características da região à qual a propriedade pertence, sendo necessário um módulo que corresponda às condições regionais, para que uma simulação realística possa ser obtida. Foram tratadas as variáveis elementares mais necessárias aos gestores: variações de temperatura, quantidade de precipitação, umidade do ar e horas de insolação por dia. Destaca-se que os anos com ocorrências de *El Niño* e *La Niña* foram analisados separadamente, devido à grande influência dos mesmos nas condições meteorológicas da América do Sul. A fonte de dados primária para a análise dessas variáveis foi o BDMEP – Banco de Dados Meteorológicos para Ensino e Pesquisa, que possui dados meteorológicos, em forma digital, de séries históricas das várias estações meteorológicas da rede de estações do INMET. As análises foram feitas com R afim de identificar distribuições de probabilidades para representar os dados climáticos citados. Após analise, foi implementado um gerador de valores aleatórios para as distribuições necessárias, afim de se gerar dados condizentes com a realidade, posteriormente utilizado na implementação do módulo climático. O módulo climático implementado foi integrado ao jogo SIMCOW, tornando assim, o jogo mais realista quando comparado a sua versão original.

Palavras-chave: Simulação. Modelagem. Meteorologia. Climatologia. R. Unity.

ABSTRACT

This paper presents the design of a climate module for a serious game format farm management simulator, built with a modular architecture that makes use of the object-oriented paradigm on the Unity game development engine. Meteorological and climatological aspects directly influence vegetation growth and animal behavior. The management of a rural property is thus directly linked to the characteristics of the region to which the property belongs, requiring a module that corresponds to regional conditions, so that a realistic simulation can be obtained. The most necessary elementary variables for managers were treated: temperature variations, amount of precipitation, air humidity and hours of sunshine per day. It is noteworthy that the years with occurrences of El Niño and La Niña were analyzed separately, due to their great influence on South American weather conditions. The primary data source for the analysis of these variables was the BDMEP - Banco de Dados Meteorológicos para Ensino e Pesquisa, which has weather data, in digital form, from historical series of the various weather stations of INMET's network of stations. The analyzes were made with R in order to identify probability distributions to represent the climate data cited. After analysis, a random value generator was implemented for the necessary distributions in order to generate data consistent with reality, later used in the implementation of the climate module. The implemented weather module has been integrated into the SIMCOW game, thus making the game more realistic compared to its original version.

Keywords: Simulation, Modeling, Meteorology, Weather, R, Unity.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANA	Agência Nacional de Águas
ANEEL	Agencia Nacional de Energia Elétrica
BDMEP	Banco de Dados Meteorológicos para Ensino e Pesquisa
BSD	Berkeley Software Distribution
EMBRAPA	Empresa Brasileira de Pesquisa Agropecuária
ENOS	El Niño-Oscilação Sul
FEE	Fundação de Economia e Estatística
FEPAGRO	Fundação Estadual de Pesquisa Agropecuária
IBGE	Instituto Brasileiro de Geografia e Estatística
INEP	Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira
INIA	Instituto Nacional de Investigaciones Agrícolas
INMET	Instituto Nacional de Meteorologia
MARN	Ministerio del Ambiente y los Recursos Naturales
MWC	Mutiply-with-carry
SAF/MDA	Secretaria de Agricultura Familiar do Ministério do Desenvolvimento Agrário
UNIPAMPA	Universidade Federal do Pampa

SUMÁRIO

1 INTRODUÇÃO

1.1 Jogos sérios e a gestão agropecuária

Jogos sérios são jogos de qualquer natureza usados para fins educacionais ou de treinamento. Seu propósito é dar apoio aos processos de ensino e aprendizagem, por meio do desenvolvimento de competências e habilidades, com vistas à formação do aluno ou profissional. Outro objetivo dos jogos sérios é de motivar os aprendizes, contribuindo na assimilação de conceitos (MICHAEL; CHEN, 2005).

Além do aspecto lúdico associado ao aprendizado por meio de jogos, existe a possibilidade de o jogo ser usado como um simulador de eventos do mundo real (MASTROCOLA, 2012). Simulações são ferramentas de desenvolvimento de conhecimento, usadas sempre que limitações de espaço, tempo ou utilização física de algo seja impraticável. Em investigações científicas, a realidade é representada por meio de *modelos* (LAW, 2007) que são simplificações da realidade capazes de capturar a essência do que se deseja representar ou investigar. Modelos podem ser construídos fisicamente (como maquetes ou estruturas menores) ou representados em algum domínio matemático; nesse último caso, podem ser resolvidos analiticamente ou simulados, quando sua resolução analítica não é possível ou não é trivial (TEIXEIRA; FERREIRA, 2018).

Sistemas de produção agropecuária, envolvendo todas as questões solo-planta-animal e ainda as questões econômicas relevantes, podem ser investigados por meio de modelos e simulação: muitas vezes não é viável testar teorias no sistema real, visto que os resultados podem levar ao colapso do sistema. A quantidade de variáveis envolvidas e seu domínio de valores, da mesma forma, não são passíveis de reprodução na realidade, visto que não há um único controle sobre as condições da realidade. Dessa forma, a melhor maneira de investigar o resultado das ações em um sistema produtivo com essas características é por meio da modelagem matemática e posterior simulação do modelo.

A Fundação de Economia e Estatística (FEE) analisou dados disponibilizados pelo Instituto Brasileiro de Geografia e Estatística (IBGE) sobre a agricultura familiar no Brasil (GRANDO, 2011) e observou que 84,4% da totalidade de propriedades agrícolas no Brasil pertencem a grupos familiares. Porém, essas propriedades representam apenas 24,3% da área destinada ao agronegócio.De acordo com a Secretaria de Agricultura

Familiar do Ministério do Desenvolvimento Agrário – SAF/MDA no Brasil são 13,8 milhões de pessoas em cerca de 4,1 milhões de estabelecimentos familiares. Cerca de 60% dos alimentos consumidos pela população brasileira e 37,8% do valor bruto da Produção Agropecuária são produzidos por agricultores familiares (OES, 2011).No Rio Grande do Sul, 85,7% dos estabelecimentos pertencem a agricultura familiar, ocupando 30,5% da área agrícola estadual.

Dados do Censo agropecuário 2017 divulgados pelo IBGE mostram que os homens são responsáveis pela direção de 81,7% dos estabelecimentos. Em 2017 apenas 5% dos diretores de propriedades rurais estavam com menos de 30 anos, 60% entre 30 e 60 anos e 35% com mais de 60 anos. Os dados mostram também que apenas 5,85% dos diretores possuem ensino superior. Observa-se, porém, que formação em Administração não prepara o gestor para as questões que envolvem conhecimentos do meio agropecuário; quando a formação superior dá-se na área de Ciências Agrárias, ela não prepara para as questões referentes à administração de propriedades.

No geral, as técnicas utilizadas na agricultura e pecuária provêm de saber empírico, muitas vezes passado de pais para filhos através das gerações, com pouca ou nenhuma fundamentação teórica.

Jogos sérios podem auxiliar gestores e seus filhos a adquirirem novos conhecimentos e experiências. D'IPOLITTO (2012) realizou uma análise sobre jogos educativos focados em gestão de negócios e em educação empreendedora, o mesmo destaca que jogos focados em gestão permitem a novos e futuros empreendedores vivenciarem as decisões típicas enfrentadas pela equipe gestora de uma pequena ou média empresa, enfrentando as incertezas de tomar decisões em mercados disputados por vários concorrentes e que possuem variáveis não controladas influenciando diretamente na gestão.

1.2 SIMCOW

Do contexto da produção agropecuária no Rio Grande do Sul surgiu a proposta do jogo SIMCOW, inicialmente apresentada em (ROBAINA; TEIXEIRA; FERREIRA, 2017). O jogo tem como objetivo operar como um simulador de gestão de propriedades rurais no formato de jogo sério. O jogador não será exposto diretamente a conteúdos a serem aprendidos, mas sim a situações-problema que proporcionarão aprendizagem por meio da experiência. Para o mesmo, foi desenvolvida uma arquitetura modular

apresentada em (ROBAINA; FERREIRA, 2018), desta forma, diferentes módulos podem ser adicionados, modificados ou retirados do jogo conforme demanda. Nesta arquitetura existem módulos específicos, chamados módulos externos, os quais tem a função de atribuir comportamentos realísticos a elementos do jogo, tais como: clima, mercado de produtos agropecuários, crescimento animal e vegetal. Este tipo de abordagem aumenta a flexibilidade de modificações e incrementos do jogo, uma vez que os módulos podem ser customizados para diferentes finalidades com fraca interferência entre eles. O grau de fidelidade desses módulos determina diretamente a qualidade final da simulação. Na implementação inicial do jogo SIMCOW (ROBAINA, 2018), o mesmo possuía módulos genéricos e sem fidelidade a dados reais, apenas com o objetivo de testes da arquitetura. O módulo climático em especifico possuía de forma estática os dados referentes ao ano de 2017, ou seja, ao longo dos anos no jogo, os dados climáticos eram sempre os dados de 2017 . Com a integração do módulo desenvolvido neste trabalho, os dados climáticos são gerados de forma dinâmica.

O *Game Design Document*(GDD) (ROBAINA, 2018) contendo os detalhes de temática, mecânicas, plataformas, fluxo de jogo, entre outros encontra-se no Anexo A.

1.3 Produção agropecuária e a meteorologia

A importância da meteorologia para a atividade agropecuária é notória: plantio, colheita, manejo de animais e vegetação estão diretamente relacionados com as condições meteorológicas. Níveis de precipitação, horas e intensidade de insolação, variações de temperatura, umidade do ar, velocidade dos ventos, etc. determinam diretamente os resultados de desenvolvimento da vegetação e do comportamento dos animais. As características locais das condições meteorológicas são determinantes no processo de tomada de decisão dos produtores. Locais com estações bem definidas possuem alterações maiores de temperatura ao longo do ano enquanto que outras regiões podem apresentar variações bem menores (PRIMACK; RODRIGUES, 2006). A gestão de uma propriedade rural está, assim, diretamente ligada às características da região à qual a propriedade pertence, sendo necessário que qualquer conhecimento usado em uma simulação corresponda às condições regionais, para que uma simulação realística possa ser obtida.

Condições meteorológicas de locais distantes podem, contudo, afetar as condições meteorológicas locais. Ainda que obedecendo a certos padrões ao longo dos anos,

dados climáticos bem estabelecidos – como a alteração da temperatura nas águas do Oceano Pacífico, que causa a ocorrência dos conhecidos fenômenos *El Niño* e *La Niña* (PHILANDER, 1985). As condições meteorológicas podem ser representadas por distribuições de probabilidade e nisto existe sempre um componente aleatório nessas condições. Dessa forma, não é desejável uma modelagem fixa ou completamente aleatória, visto que nenhuma delas se aproxima da realidade. A abordagem de modelagem meteorológica precisa variar sobre características esperadas a partir de dados históricos referentes às estações do ano e aos fenômenos meteorológicos esperados.

1.4 Objetivos

O objetivo deste trabalho é desenvolver um módulo de geração de condições climáticas para ser usado em um jogo de simulação de gestão de propriedades rurais.

São definidos como objetivos específicos deste trabalho:

- Entender os processos de previsão de tempo e as variáveis envolvidas, baseado em séries históricas;
- Aprofundar os conhecimentos de distribuições de probabilidades e sua aplicação em questões climáticas;
- Entender os aspectos do clima do Rio Grande do Sul, com ênfase à região de Bagé, para que o módulo entregue seja o mais fiel possível às características climáticas da região;
- Projetar um módulo que possa ser facilmente alterado para uso de dados de outras regiões.
- Implementar o módulo sobre o motor (engine) de desenvolvimento de jogos Unity.

O restante do texto deste trabalho está organizado como se segue: o Capítulo 2 apresenta o referencial teórico desta pesquisa. A seção 2.1 aborda conceitos de meteorologia e climatologia, fenômenos meteorológicos e a importância dos mesmos para a agropecuária. A Seção 2.2 aborda aspectos gerais sobre simulações e jogos sérios que fazem uso das mesmas para reproduzirem eventos do mundo real. A Seção 2.3 aborda modelos probabilísticos, testes de aderência de amostras a distribuições de probabilidade e suas aplicações a questões climáticas. A seção 2.4 apresenta conceitos sobre a geração de valores aleatórios, com enfase a geração de números aleatórios que representem determinadas distribuições de probabilidade. Por fim, a Seção 2.5 apresenta

os trabalhos correlatos a este.

O Capítulo 3 apresenta a metodologia utilizada para o desenvolvimento deste trabalho. Inicialmente é feita a caracterização da pesquisa desenvolvido, é apresentado um diagrama de metodologia e em seguida são descritos os passos realizados neste trabalho.

O Capítulo 4 relata o desenvolvimento deste trabalho. A Seção 4.1 descreve o método utilizado para importação e padronização dos dados climáticos no RStudio. A Seção 4.2 apresenta o procedimento realizado para analisar dados e os seus resultados obtidos. A Seção 4.3 descreve o gerador de valores aleatórios implementado. A Seção 4.4 descreve a utilização do gerador de valores aleatórios na implementação das variações climáticas no jogo sério SIMCOW. A Seção 4.5 descreve o tipo de teste que foi utilizado em cada etapa do desenvolvimento.

O Capítulo 5 apresenta as conclusões do trabalho.

2 REFERENCIAL TEÓRICO

2.1 Meteorologia e climatologia

Segundo SILVA (2006) a meteorologia é a ciência que estuda a atmosfera terrestre. Seus aspectos mais tradicionais e conhecidos são a previsão do tempo e a climatologia. O tempo pode ser definido como o estado da atmosfera em determinado instante e lugar. O clima é tradicionalmente definido como um conjunto de condições normais que dominam uma região, obtidas das médias das observações durante um certo intervalo de tempo. Contudo, variações e condições extremas do tempo também são importantes para caracterizar o clima de uma região. Portanto, a climatologia estuda os fenômenos atmosféricos do ponto de vista de suas propriedades estatísticas para caracterizar o clima em função da localização geográfica, estação do ano, hora do dia, etc. A longo prazo, é o clima que determina as condições de uma região e sua vegetação natural; a curto prazo, é a meteorologia que condiciona a dispersão de poluentes e as atividades da agricultura. (SILVA, 2006) ainda lista as seguintes variáveis como descritores do tempo: a temperatura do ar, a umidade do ar, a pressão do ar, a velocidade e a direção do vento, o tipo e a quantidade de precipitação e o tipo e a quantidade de nuvens.

Segundo Gonçalves e Back (2018) dentre as variáveis climáticas, a precipitação, assim como a temperatura, são os elementos que atuam de maneira mais direta nas alterações ambientais. Variações da temperatura afetam tanto a disposição dos animais para alimentarem-se quanto o desenvolvimento foliar da vegetação (PRIMACK; RODRIGUES, 2006). A variabilidade da precipitação condiciona os ciclos agrícolas e outras atividades humanas. Além disso, efeitos inesperados da precipitação – como períodos com excessos de chuvas ou períodos longos de seca – em ambientes rurais implicam na perda parcial ou total de safras, comprometendo assim o mercado de produtos agrários. Isto é reforçado por Cera e Ferraz (2015) que afirma que a produção agropecuária do Rio Grande do Sul tem o clima como principal fator responsável por perdas na safras e, consequentemente, diminuição de lucros. Nascimento et al. (2018) estudaram o efeito das variações de temperatura sobre as plantações de soja e milho, duas das principais culturas plantadas no Brasil, concluíram que temperaturas mínimas abaixo de 10^{o}C e máximas acima de 30^{o}C prejudicam o desenvolvimento da soja, assim como mínimas abaixo de 14^{o}C e máximas acima de 40^{o}C prejudicam o desenvolvimento do milho.

O fenômeno *El Niño-Oscilação Sul* (ENOS) é o fenômeno mais relevante para a variabilidade climática interanual em escala global (KAYANO et al., 2016) . O fenômeno é chamado de *El Niño* quando as águas do Pacífico tropical estão quentes e o Índice de Oscilação Sul esta negativo e como *La Niña* quando as águas do Pacífico tropical estão frias e o Índice de Oscilação Sul positivo. Segundo Fontana e Berlato (1997) o índice de oscilação sul (IOS) é um fenômeno de grande escala que é caracterizado pela diferença de pressão padronizada entre Darwin (Austrália) e Taiti (Polinésia Francesa) e é considerado positivo quando a pressão está maior no Taiti e negativo quando a pressão está maior em Darwin. O comportamento da temperatura da superfície das águas do Oceano Pacífico tropical em sua parte central e junto à costa oeste da América do Sul associado aos campos de pressão representados pelo Índice de Oscilação Sul, altera o padrão de circulação geral da atmosfera. Com isso, acaba influenciando no clima de diferentes regiões do mundo e sendo o responsável pelos desvios em relação ao clima considerado normal (CUNHA et al., 2011).

O ENOS afeta o clima de cerca de 20 regiões no mundo, incluindo algumas regiões do Brasil: a parte norte da Região Nordeste, o leste da Amazônia (na faixa tropical) e a Região Sul (na faixa extratropical). As anomalias climáticas de maior impacto são as relacionadas com o regime de chuvas, embora as temperaturas dessas regiões também possam ser afetadas. As anomalias decorrentes do El Niño e La Niña usualmente atingem as mesmas regiões nos mesmos períodos do ano ou com pouca defasagem, ou seja, naquelas regiões onde em anos de El Niño há excesso de chuvas, nos anos de La Niña pode ocorrer seca. Segundo Cunha et al. (2011) a precipitação pluvial é uma das principais causas de flutuação no rendimento das culturas e da produção agrícola, sendo que, na Região Sul do Brasil, a variabilidade ao longo dos anos das chuvas é influenciada diretamente pelo fenômeno ENOS. Em geral, observa-se excesso de chuvas nos anos de El Niño e estiagem em anos de La Niña. A influência do ENOS é observada durante todo o período de atuação do evento, porém, há duas épocas do ano que são especialmente afetadas, sendo a primeira no ano inicial do fenômeno, entre outubro e dezembro, ou seja, entre a primavera e começo do verão e a segunda no ano seguinte, entre abril e junho, abrangendo o final do outono e o começo do inverno. Assim, nessas épocas, as chances de chuvas acima do normal são maiores em anos de El Niño e chuvas abaixo do normal, em anos de La Niña. As fases extremas de Oscilação Sul provocam alterações nos totais mensais e sazonais de precipitação pluvial na Região Sul do Brasil, sendo a primavera a estação que sofre o maior impacto. O El Niño apresenta primaveras mais chuvosas e o La

Niña primaveras mais secas.

No que se refere à agricultura, as duas épocas afetadas são relevantes por estarem relacionadas a períodos específicos de safras. No período de outubro a dezembro, as principais culturas de primavera-verão cultivadas no Rio Grande do Sul encontram-se na fase de estabelecimento e desenvolvimento. Precipitação pluvial muito abaixo da média climatológica compromete o rendimento final das culturas, ao passo que precipitação ligeiramente acima da média climatológica pode favorecê-las. O segundo período de influência do ENOS (abril e maio) coincide com a maturação e colheita das principais culturas de primavera-verão. Nesse período, a precipitação pluvial acima da média climatológica prejudica, ao passo que precipitação abaixo da média climatológica favorece (FONTANA; BERLATO, 1997) o processo.

A atipicidade dos valores das variáveis estabelecidas, resultante de fenômenos meteorológicos como o ENOS, deve ser levada em consideração no projeto do módulo climático, tratando assim os anos com ocorrência desses eventos de maneira separada. De outra forma, os valores gerados pela distribuição de probabilidade escolhida para a variável podem ser irreais em anos ditos normais, assim como em anos que sofreram os efeitos de tais fenômenos.

A Tabela 1 apresenta as últimas ocorrências de ambas as fases do fenômeno ENOS. É possível notar que o fenômeno ocorre com bastante frequência em ambas as fases, nota-se que desde o ano 2000, ambos El Niño E La Niña ocorreram por quatro vezes, porém, analisando todo o conjunto, é possível notar que a ocorrência do El Niño foi mais frequente desde a década de 1960. O INPE calcula também a intensidade das alterações climáticas em cada ocorrência. Nota-se que a intensidade moderada predomina em ambas as fases, porém, enquanto ocorrências de El Niño com intensidade forte predominam sob as fracas, o contrário ocorre com La Ninã.

Tabela 1 – Ocorrências de El Niño e La Niña

El Niño	Intensidade	La Niña	Intensidade
1963-1964	Moderada	1933-1934	Moderada
1965-1966	Forte	1937-1938	Fraca
1968-1969	Fraca	1938-1939	Fraca
1969-1970	Moderada	1942-1943	Forte
1972-1973	Forte	1949-1950	Forte
1976-1977	Moderada	1954-1955	Fraca
1979-1980	Moderada	1955-1956	Moderada
1982-1983	Forte	1967-1968	Fraca
1986-1987	Moderada	1970-1971	Fraca
1987-1988	Forte	1973-1974	Moderada
1991-1992	Forte	1975-1976	Moderada
1992-1993	Fraca	1988-1989	Moderada
1997-1998	Forte	1998-1999	Fraca
2002-2003	Moderada	1999-2000	Moderada
2006-2007	Moderada	2007-2008	Moderada
2009-2010	Moderada	2010-2011	Moderada
2015-2016	Forte	2017-2018	Moderada

Fonte: INPE (2019)

2.2 Simulação

Segundo Law (2007) um processo de interesse é usualmente chamado de *sistema* e, para estudá-lo cientificamente, são feitas uma série de suposições sobre como ele funciona. Essas suposições resultam em um modelo, que usualmente é descrito em linguagem matemática ou de relacionamentos lógicos, por meio diagramas ou outras formas de representação visual. O modelo tem como objetivo ganhar conhecimento sobre

como o sistema se comporta (ALTIOK; MELAMED, 2010). Se os relacionamentos que compõem o modelo são simples, pode ser possível usar métodos matemáticos como álgebra, cálculo diferencial ou integral e teoria das probabilidades para se obter resultados exatos do sistema de interesse. Sistemas que podem ser resolvidos por teorias matemáticas são ditos possuir solução analítica. Porém, a maioria dos sistemas do mundo real são muito complexos para que modelos realísticos dos mesmos sejam avaliados de forma analítica, o que leva a necessidade de estudar esses modelos por meio de simulações.

Zeigler, Kim e Praehofer (2000) definem simulação como um modelo de um conjunto de problemas ou eventos que podem ser usados com objetivo de ajudar alguém a obter novos conhecimentos e habilidades. Simulação é também um dos métodos mais utilizados para avaliação de técnicas científicas e gerenciais. Law (2007) ainda cita diversas áreas onde simulações provaram ser ferramentas úteis e poderosas: projeto e análise de fábricas antes de sua construção, avaliação de sistemas de armas militares, determinação de requisitos de hardware para redes de comunicações, determinação de requisitos de hardware e software para sistemas computacionais, projeto e análise de sistemas de transportes como aeroportos, autoestradas, portos, metrôs e análise de cadeias de fornecimento. Em (ALTIOK; MELAMED, 2010), a modelagem é definida como o processo de construção de representações simplificadas de sistemas reais complexos, com o objetivo de prover previsões de comportamento do sistema de acordo com métricas de interesse. Modelagens exigem abstrações (no sentido de que elementos de interesse devem ser categorizados de forma geral) e simplificações (uma vez que nem todos os detalhes de um sistema real podem ou devem fazer parte de um modelo).

Diversos jogos sérios reproduzem eventos do mundo real por meio de simulações. Em Reis e Rocha (2017) é realizada uma análise em cima do *CardioSim*, um jogo/simulador brasileiro para ensino de ressuscitação cardiorrespiratória, no qual são simulados os principais passos para realizar a ressuscitação com sucesso, sendo que o jogador define a ordem de execução destes passos e tem como retorno as consequências de possíveis erros. Paravisi e Amory (2017) estudam o resgate de pessoas em multidões em situações de emergência através de simulações; o jogo explora um dos principais desafios para os responsáveis por estes resgates: o comportamento humano e a imprevisibilidade de como as multidões se comportarão em situações emergenciais, comparando possíveis formações de grupos. Em Silva e Alves (2017) é descrito o *Ocean Simulator*: uma simulação de vida marinha que, utilizando parâmetros biológicos como cadeia alimentar

e pirâmide de energia, possa-se obter como resultado interações entre personagens que imitem os comportamentos de animais marinhos, objetiva a aprendizagem de informações importantes sobre o ecossistema marinho e seus funcionamentos. Peletti (2015) apresenta o Micro Dentista, que busca simular de maneira divertida aspectos relacionados a saúde bucal mostrando as doenças bucais e como são adquiridas, métodos de prevenção de doenças bucais e métodos de combate e cura a essas doenças.

2.3 Modelos probabilísticos

A Teoria das Probabilidades permite a formalização de incertezas com rigor matemático. O espaço amostral é o conjunto formado por todos os resultados possíveis de um experimento aleatório. Um experimento Ω possui um espaço amostral associado. Uma função X, que associa a cada elemento $\omega \in \Omega$ um número real, $X(\omega)$, é denominada variável aleatória. Ou seja, variável aleatória é um característico numérico do resultado de um experimento (MEESTER, 2008).

Uma distribuição de probabilidade é um modelo de descrição probabilística do conjunto de todos os valores que uma variável aleatória pode assumir, justamente com a frequência esperada de cada valor possível para a variável Spiegel, Schiller e Srinivasan (2016). Na análise de uma variável aleatória é relevante saber: o tipo de distribuição de probabilidade da variável, a função de probabilidade da variável e os parâmetros da distribuição. Os parâmetros da distribuição podem ser exemplificados através das distribuições Normal e de Cauchy, a Normal é caracterizada através de dois parâmetros: média e desvio padrão. Enquanto a distribuição de Cauchy é caracterizada por α e β, respectivamente localização do pico da distribuição e escala que representa a dispersão da distribuição.

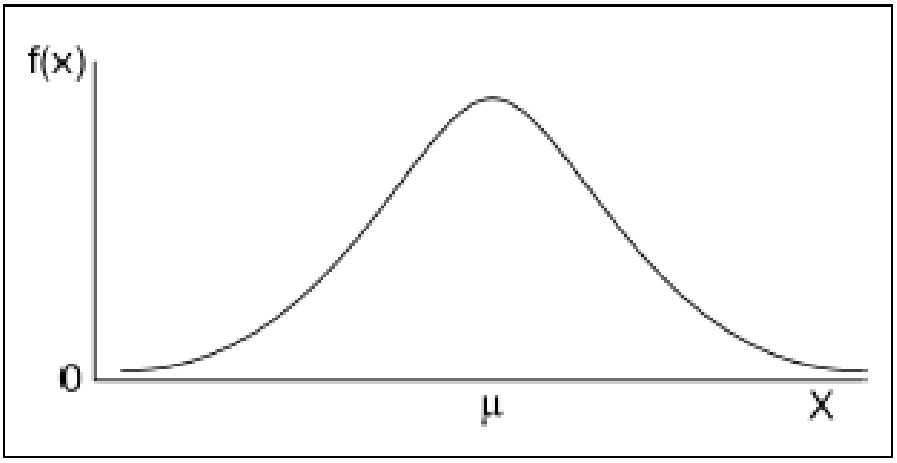
Os modelos descritores do comportamento probabilístico podem ser de variáveis discretas e contínuas. São discretas todas as variáveis cujo espaço amostral é enumerável (finito ou infinito) (MANN, 2008). As principais distribuições discretas são a distribuição de Bernoulli, distribuição Binomial, distribuição Uniforme, distribuição Hipergeométrica e distribuição de Poisson.

São contínuas todas as variáveis cujo espaço amostral é infinito não enumerável. Assim, uma variável aleatória contínua pode assumir qualquer valor num intervalo [a; b] ou no intervalo (-∞; + ∞). As principais distribuições contínuas são a distribuição Normal, distribuição de Cauchy, distribuição Qui-quadrado, distribuição Log-Normal, distribuição

Exponencial, distribuição Gama, distribuição Beta, distribuição de Gumbel e distribuição de Weibull.

A Figura 1 apresenta o formato de uma distribuição normal, onde μ é a média e σ o desvio padrão.

Figura 1 – Distribuição normal



Fonte: Spiegel, Schiller e Srinivasan (2016)

A Distribuição Normal é uma das distribuições de probabilidade que tem maiores aplicações na Estatística. É também conhecida como Distribuição de Gauss-Moivre-Laplace. Além de descrever uma série de fenômenos físicos e financeiros, a Distribuição Normal possui grande uso no campo da Estatística Inferencial, sendo imprescindível para o desenvolvimento da amostragem, estimação por intervalo e testes de hipóteses. É inteiramente descrita por seus parâmetros média e desvio padrão (ou, de forma equivalente, a variância), ou seja, conhecendo-se estes, consegue-se determinar qualquer probabilidade em uma Distribuição Normal.

Testes de aderência, como o Qui-quadrado, Kolmogorov-Smirnov, Lilliefors, Shapiro-Wilk e Cramer-von Mises, servem para comparar as probabilidades empíricas de uma variável com as probabilidades teóricas estimadas pela função de distribuição em teste, verificando se os valores da amostra podem razoavelmente ser considerados como provenientes de uma população com aquela distribuição teórica (RAZALI; WAH et al., 2011). Nos testes de aderência, a hipótese nula (H_0) admite que a distribuição seja a especificada (Normal, Log-Normal, Gama e outras), com os seus parâmetros estimados com base nos dados amostrais (SPIEGEL; SCHILLER; SRINIVASAN, 2016).

O teste de Lilliefors (CAMPOS, 1983) é mais eficaz que o teste de Kolmogorov-Smirnov, porém é específico para verificar a aderência dos dados amostrais

à distribuição normal. Os testes de Shapiro-Wilk e Cramer-von Mises também diferenciam-se quanto à sua eficácia. O teste de aderência de Qui-quadrado apresenta limitações. Por exemplo, a frequência de uma classe não pode ser inferior a cinco e os dados são agrupados em classes perdendo informações, o que não ocorre no teste de Kolmogorov-Smirnov, que além de poder ser realizado com os dados agrupados, pode também ser realizado com os dados isoladamente, sendo normalmente mais eficiente que o Qui-quadrado em pequenas amostras, ou seja, menos de 30 observações. O teste de Kolmogorov-Smirnov é baseado no módulo da maior diferença entre a probabilidade observada e a estimada, que é comparada com um valor tabelado de acordo com o número de observações da série sob teste (FILHO; MATZENAUER; TRINDADE, 2004).

Diversos autores vêm estudando as aplicações de distribuição de probabilidades em questões climáticas. Filho, Matzenauer e Trindade (2004) verificaram o ajuste das séries de dados de radiação solar global média decendial de 22 municípios do Estado do Rio Grande do Sul às funções de distribuições de probabilidade Normal, Log-normal, Gama, Gumbel e Weibull. Foram utilizados dados do período entre 1956 e 2003 disponíveis no banco de dados do laboratório de agrometeorologia da Fundação Estadual de Pesquisa Agropecuária (FEPAGRO). Foi aplicado o teste de aderência de Kolmogorov-Smirnov à série de dados de radiação solar global média decendial, para verificar o ajuste dos dados às distribuições. O teste de aderência de Kolmogorov-Smirnov revelou a distribuição Normal como a mais adequada ao estudo para representar radiação solar global média decendial, bastando estimar os parâmetros desta distribuição (média e desvio-padrão) para os cálculos de probabilidade dentro de limites de intervalos desejados.

Em Lyra et al. (2006) foi realizada uma pesquisa com o objetivo de determinar regiões homogêneas, de acordo com a sazonalidade da precipitação mensal e a distribuição de probabilidade que melhor se ajusta a essas regiões no Estado de Táchira, Venezuela. Foram utilizados valores da precipitação pluvial mensal de 25 estações climatológicas do estado obtidas pelo *Ministerio del Ambiente y los Recursos Naturales* (MARN) e pelo *Instituto Nacional de Investigaciones Agrícolas* (INIA). As estações apresentam séries históricas entre 24 e 62 anos. No agrupamento dos meses com média da precipitação mensal similar, foi aplicado a análise de agrupamento (*cluster*). Utilizou-se o método hierárquico aglomerativo de Ward (WARD, 1963). Pela análise de agrupamento, foram observados dois períodos bem diferenciados para chuva mensal, sendo um correspondente aos meses secos (janeiro, fevereiro, março e dezembro) e outro

formado pelos úmidos (de abril a novembro), o período úmido apresenta ainda uma subdivisão que separa os meses de transição (abril e novembro). Foi concluído que no período seco, a distribuição Exponencial era a que mais se aproximava da realidade, enquanto a que melhor representou o período úmido foi a distribuição Normal.

Silva et al. (2007) analisaram a distribuição da quantidade diária de precipitação e do número de dias com chuva e determinaram a variação da probabilidade de ocorrência de precipitação diária, durante os meses do ano, em Santa Maria, RS. Os dados de precipitação pluvial utilizados foram obtidos no Banco de Dados Meteorológicos para Ensino e Pesquisa (BDMEP), abrangendo o período entre 1968 e 2004. Foram realizados os testes de Anderson-Darling, Cramérvon Mises, Qui-quadrado e Kolmogorov-Smirnov (D'AGOSTINO, 1986). Concluíram que as funções de distribuição de probabilidades que melhor se ajustaram aos dados diários de precipitação, foram Gama e Weibull, não havendo diferença entre as probabilidades calculadas por essas funções.

Sansigolo (2008) analisou o ajuste de diferentes distribuições de extremos às séries de precipitação diária, temperaturas máximas e mínimas absolutas no período de 1917 a 2004 e a velocidade instantânea do vento no período de 1956 a 2006 em Piracicaba, SP. Testes de Kendall (KENDALL et al., 1946) foram utilizados para evidenciar a existência de eventuais tendências nas séries. Testes de Fisher (FISHER, 1935) também foram aplicados às séries para verificar a existência de periodicidades significativas. Os ajustes e a seleção das melhores distribuições teóricas foram feitos por testes do Qui-Quadrado, que compara o efetivo observado e o teórico esperado em intervalos discretos e Kolmogov-Smirnov (D'AGOSTINO, 1986), que compara as distribuições empíricas acumuladas com as teóricas. A distribuição que melhor se ajustou às temperaturas máximas absolutas e precipitação foi a de Gumbel enquanto a que se ajustou melhor às temperaturas mínimas foi a distribuição Normal, as velocidades instantâneas máximas anuais do vento foram melhor ajustadas pela distribuição de Weibull.

(BACK, 2001) analisou dados de cem postos pluviométricos do estado de Santa Catarina com período de dados entre doze e setenta anos, pertencentes a rede de estações pluviométricas da Agencia Nacional de Energia Elétrica (ANEEL). O trabalho teve como objetivo selecionar uma distribuição de probabilidade para a estimativa da precipitação máxima anual. Para a verificação do ajuste da distribuição de probabilidade foi utilizado o teste de Kolmogorov-Smirnov e para escolher a melhor distribuição, foi utilizado o critério do menor erro padrão. A distribuição de Gumbel apresentou o melhor ajuste em

60% das estações analisadas e em 93% das estações com menos de 20 anos de dados. Para as séries com baixa assimetria a distribuição Log-normal com três parâmetros apresentou o melhor ajuste, enquanto que para séries com alta assimetria,a distribuição Log-Pearson, seguida da distribuição Log-normal com dois parâmetros, foram as que forneceram os melhores ajustes.

Foi observado que o teste de Kolmogorov-Smirnov é um dos testes constantemente utilizados para encontrar a melhor distribuição de probabilidade em dados climáticos.

2.4 Geração de valores aleatórios

Para gerar realizações de uma distribuição de probabilidade específica, como a Distribuição Normal, por exemplo, precisamos gerar números aleatórios. Isso não pode ser realizado por máquinas, pois, na verdade, qualquer sequência produzida por uma máquina é na realidade uma sequência previsível. Daí, a denominação de sequência de números pseudoaleatórios (SOUZA; JR, 2011). Uma sequência de números será considerada "aleatória" do ponto de vista computacional se o programa que a gerar for diferente e estatisticamente não correlacionado com o programa que a usará (FERREIRA, 2007).

(ZAFALON; JR, 2006) enfatiza que para uso de geradores de valores aleatórios em simuladores é preciso que o mesmo seja capaz de gerar números que obedeçam diferentes funções de distribuição de probabilidade. Nesses casos é preciso ainda que esses geradores sejam independentes, para que não ocorram interferências entre os números gerados em cada distribuição. Esse último problema leva ao gerador da distribuição uniforme, que é a base dos demais geradores.

Vários métodos foram desenvolvidos e podem ser implementados em algoritmos. Esses métodos são chamados de geradores de números aleatórios. Segundo Souza e Jr (2011) os principais métodos utilizados para a geração da distribuição uniforme são: Linear Congruente, Multiply-With-Carry, Lagged Fibonacci e Mersenne Twister.

O método Linear Congruente é baseado na seguinte relação de recorrência (SOUZA; JR, 2011):

$$x_{n+1} = (ax_n + c) \mod m, n \geq 0$$

Nessa relação, m é chamado de módulo2, a de multiplicador e c de incremento. Se esses valores forem escolhidos de forma errada, o período da sequência de números

aleatórios pode ser drasticamente afetado. O período será sempre maior que m, logo, para aumentar o período deve-se ter o maior valor de m possível. Caso os parâmetros a, c e m sejam escolhidos de forma adequada, o algoritmo, apesar de simples, funciona conforme o esperado.

O método *Mutiply-with-carry* (MWC), criado por George Marsaglia, é uma variação do Método Linear Congruente. A principal vantagem do método MWC é que ele gera sequências muito rapidamente e com períodos variando entre 2^{60} e $2^{2.000.000}$. Os geradores MWC costumam se comportar melhor que os outros geradores em testes de aleatoriedade. Na sua forma mais comum, um gerador MWC necessita de uma base b, um multiplicador a, um conjunto de valores aleatórios de sementes e um valor inicial para c. Uma sequência MWC é então uma sequência de pares x_n:

$$x_n = (ax_{n-r} + c_{n-1}) \mod b$$

E c_n é determinado por:

$$c_n = \frac{ax_{n-r} + c_{n-1}}{b}$$

Em comparação ao gerador Congruente Linear, a principal diferença entre eles é que no método MWC o parâmetro c varia ao longo de cada iteração.

O método MWC, com valores adequados para os parâmetros, passa em testes estatísticos que os geradores congruentes lineares não passam (SOUZA; JR, 2011).

Geradores Lagged Fibonacci são baseados na sequência Fibonacci, que pode ser descrita pela seguinte relação de recorrência:

$$S_n = S_{n-1} + S_{n-2}$$

A fórmula da sequência de Fibonacci mostra que cada termo da sequência é igual a soma dos seus dois termos antecessores. Já o gerador Lagged Fibonacci é caracterizado por (ROSA, 2016):

$x_n = (x_{n-j} \star x_{n-k}) \mod m, 0 < j < k$

Ou seja, uma sequência semelhante à de Fibonacci, mas onde cada termo Xn da sequência depende de valores deslocados de nj e nk. O símbolo $\star$ simboliza uma das seguintes operações: adição, multiplicação ou OU exclusivo (XOR).

O Mersenne Twister é um gerador de números pseudoaleatórios que foi desenvolvido em 1997 por Makoto Matsumoto e Takuji Nishimura. Ele é uma variação

do gerador Lagged Fibonacci.

As principais características do Mersenne Twister são o fato de possuir um período muito longo, de $2^{19.937}$˘1, além de ter sido aprovado pelos principais testes de aleatoriedade existentes ROSA (2016).

O método para gerar valores da Distribuição Normal mais antigo e conhecido é o Método de Box-Muller (BOX; MULLER, 1958), o qual produz um par de números aleatórios que seguem uma distribuição Normal padrão a partir de um par de números de uma distribuição Uniforme. Este método utiliza o fato de que a distribuição bidimensional de dois números aleatórios normais com média zero é radialmente simétrica se ambos os componentes normais tem a mesma variância (SCHUTZ, 2012). O algoritmo de Box-Muller pode ser entendido como um método no qual os números normais de sua saída representam as coordenadas no plano bidimensional. Devido ao algoritmo produzir dois números aleatórios cada vez que é executado, é comum a sua função geradora retornar o primeiro valor para o usuário e ocultar o outro valor para retornar na próxima chamada da função.

2.5 Trabalhos correlatos

Nesta seção serão apresentados trabalhos correlatos ao trabalho aqui apresentado. Foram considerados correlatos os trabalhos que envolviam jogos sérios que buscaram modelar o clima de forma realista e que divulgaram a maneira como a modelagem foi feita. Trabalhos que modelam o clima de forma realista, mas não detalham como é feito, não foram considerados, visto não haver possibilidade de comparação com a abordagem aqui desenvolvida. Também foram considerados correlatos trabalhos que implementam geradores de valores aleatórios para diferentes distribuições de probabilidade e fins de aplicação.

Little Botany é um jogo sério desenvolvido para dispositivos móveis apresentado em (JAMONNAK; CHENG, 2017). O jogo tem como objetivo ensinar jardinagem para crianças em idade escolar. Os jardins virtuais do *Little Botany* utilizam o banco de dados da empresa *Weather Underground*, que fornece serviço meteorológico comercial. Os dados são carregados com base na localidade selecionada pelo jogador e os mesmos são atualizados diariamente, sendo que a periodicidade da atualização ao longo do dia depende da localidade selecionada. O jogador possui a opção de selecionar uma data para iniciar o jogo, dentre as quais os dados estejam disponíveis para a cidade selecionada. Os

autores consideraram os seguintes atributos existentes no banco de dados relevantes para a pratica de jardinagem: horário, temperatura, umidade relativa do ar, ocorrência ou não de precipitação e sua intensidade.

O jogo Calangos (SOUZA et al., 2010) é voltado para o ensino e aprendizagem de evolução, utilizando o processo evolutivo dos lagartos como exemplo, e ecologia, inspirado nas dunas do médio Rio São Francisco, no Estado da Bahia, contexto que coloca o estudante em contato com o bioma Caatinga. O objetivo final do jogo é possibilitar ao estudante um ambiente com suficiente realismo, permitindo uma compreensão adequada dos processos ecológicos e evolutivos. O jogador controla um lagarto de uma entre três das espécies existentes na região.

A modelagem climática do jogo Calangos é descrita em (LOULA et al., 2009) e sua versão final em (SOUZA et al., 2010). A abordagem não buscou realizar uma simulação do clima que procurasse representar acuradamente todas as variáveis do caso real, porém, não se tratou de uma implementação arbitrária do clima, buscou-se aproximar o modelo da dinâmica climática da região de uma maneira que fosse suficiente para representar as relações ecológicas dos lagartos simulados com o clima, bem como para a compreensão pelos estudantes de como estas relações têm lugar. A partir da descrição do caso ecológico, os desenvolvedores determinaram que as seguintes variáveis climáticas são mais relevantes para a simulação necessária ao jogo: temperatura do ar e do solo, umidade relativa do ar e precipitação. Estas variáveis afetam, de forma direta ou indireta, características dos lagartos modelados no jogo, como a temperatura interna, o gasto energético, a energia acumulada e a hidratação. A modelagem foi feita de forma que os dados climáticos variassem temporalmente ao longo do dia, assim como com o passar dos dias e dos meses, de acordo com as mudanças climáticas relacionadas às estações do ano.

Foram analisados dados de variação diária de temperatura em regiões do interior do nordeste, através de dados disponibilizados pelo Instituto Nacional de Meteorologia (INMET) e observado que existe um padrão de variação de temperatura ao longo das 24 horas do dia. A partir do nascer do sol, aproximadamente 6 horas da manhã no conjunto de dados obtidos, a temperatura começa a se elevar rapidamente, diminuindo a velocidade de aumento por volta das 14 horas. A partir deste horário a temperatura se estabiliza e começa, então, a decrescer aos poucos, com a velocidade da queda de temperatura crescendo até chegar às 21 horas. A partir deste horário, a temperatura começa a cair linearmente até às 6 horas, iniciando-se, então, um novo ciclo. Essa observação permitiu

aos desenvolvedores criar uma função que se aproxima desse padrão e fornece para o jogo a temperatura em graus centígrados a cada hora do dia, parametrizada pela temperatura máxima e mínima do dia. Para determinar as temperaturas máximas e mínimas, utilizaram-se informações sobre séries históricas da região para cada mês do ano. Essas informações foram transportadas para o jogo como duas variáveis com distribuição Normal, com médias e desvio padrões específicos para cada mês, aproximando-se de dados históricos.

A modelagem da umidade relativa do ar para o jogo se deu de forma semelhante à modelagem da temperatura do ar, com a análise visual dos dados de variação de umidade relativa do ar através de gráficos disponíveis no site do INMET.

Para determinar os dias em que há precipitação, foram analisados dados históricos de precipitação a cada mês ao longo dos anos e foi estabelecida a média e o desvio padrão para cada mês, permitindo determinar uma variável aleatória de distribuição Normal para a precipitação mensal. Para determinar a precipitação diária em um mês, inicialmente sorteou-se a precipitação mensal e dividiu-se esse valor pelo maior valor de precipitação entre todos os meses da série histórica, obtendo assim um provável número de dias com chuva dentro deste mês. A razão entre o provável número de dias com chuva e o total de dias do mês fornece, então, uma probabilidade diária de chuva no mês em questão.

O jogo Calangos por possuir abordagem semelhante a este contribuiu com metodologias de como efetuar a análise dos dados obtidos. O jogo *Little Botany* apresentou uma abordagem que não havia sido considerada para este trabalho, apresentando assim uma maneira diferente de resolver o mesmo problema.

Zafalon e Jr (2006) implementaram na linguagem Java um gerador de números aleatórios, que atende o perfil de distribuição uniforme. A partir do gerador para a distribuição uniforme foi possível definir geradores para outras funções de distribuição. Dentre as várias funções de distribuição existentes, adotaram-se no trabalho as distribuições Normal, Exponencial, Gama e Pareto. O gerador implementado foi validado através do teste do Chi-Quadrado. O trabalho conclui que a precisão obtida com o gerador foi significativa, permitindo que se trabalhe com a adequação de cada gerador de distribuição a valores distintos de média e desvio padrão. Assim, o mesmo pode concluir que os geradores implementados são suficientemente robustos e podem ser usados na construção de ferramentas mais genéricas de simulação.

Cook (2014b) implementou um gerador de valores aleatórios na linguagem de programação C++, utilizando o algoritmo *Multiply-With-Carry* para a implementação

da distribuição Uniforme, que é usado como base para a implementação das outras distribuições de probabilidades que compõem o gerador: Beta, Cauchy, Chi-Quadrado, Exponencial, Gama, Normal, Student T e Weibull. O Gerador passou na bateria de testes *DIEHARD*(MARSAGLIA, 1996), método de validação de geradores desenvolvido pelo mesmo criador do algorítimo *Multiply-With-Carry*, George Marsaglia. O gerador de números aleatórios teve seu código disponibilizado sob a licença *Berkeley Software Distribution* (BSD), uma licença de código aberto, permitindo assim sua modificação e utilização para diversos fins.

Cook (2014b) e Zafalon e Jr (2006) seguiram metodologias semelhantes para a implementação dos seus respectivos geradores. Inicialmente implementaram a geração de amostras de uma distribuição Uniforme. Posteriormente a mesma é utilizada na implementação das demais distribuições. A mesma metodologia foi adotada no presente trabalho.

3 METODOLOGIA

Seguindo a classificação apresentada em Freitas (2013), este trabalho é classificado quanto a sua natureza como uma pesquisa aplicada pois objetiva gerar um módulo climático com aplicação prática dirigida à solução de um problema específico. Quanto aos seus objetivos, ela pode ser classificada como exploratória e descritiva, pois visa implementar distribuições estatísticas que descrevam adequadamente o clima do Rio Grande do Sul. Do ponto de vista dos procedimentos técnicos ela é classificada como quantitativa, pois será baseada em dados numéricos de clima para especificação dos modelos.

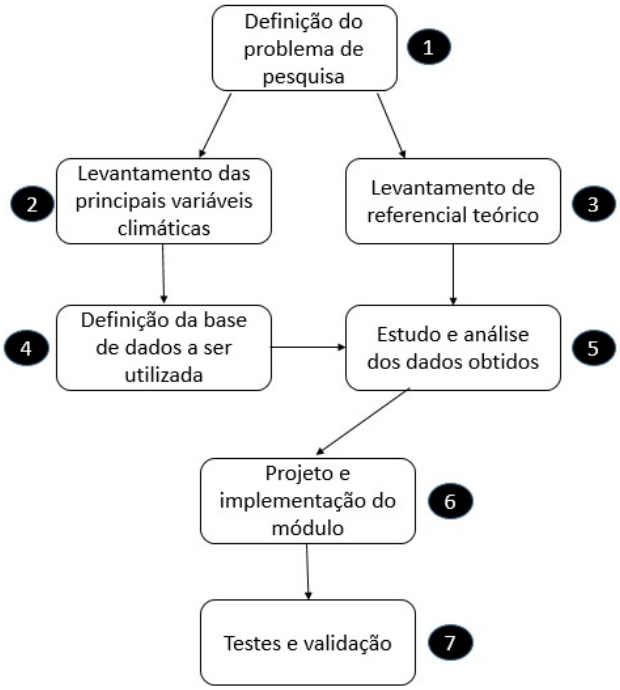
A metodologia de pesquisa orientou a definição dos procedimentos que foram executados ao longo de dois semestres letivos, dividindo as tarefas entre os componentes curriculares matriculados TCC I e TCC II. A Figura 2 ilustra as etapas necessárias para o desenvolvimento e conclusão deste trabalho, que são detalhadas a seguir.

Definição do problema de pesquisa A primeira etapa consistiu em definir o problema de pesquisa do trabalho. Devido ao papel dos aspectos climáticos e meteorológicos na agropecuária foi notada a necessidade do jogo SIMCOW (ROBAINA, 2018) possuir um módulo que tratasse exclusivamente destes aspectos, objetivando assim uma maior percepção de realidade em relação as mudanças climáticas ocorridas no jogo bem como tornar o jogo um simulador efetivo de condições reais de manejo de propriedades rurais.

Levantamento das principais variáveis climáticas A segunda etapa do trabalho consistiu no levantamento das principais variáveis climáticas a serem implementadas no jogo. A busca de informações foi realizada por meio de referencial bibliográfico, conversas com um dos especialistas em Agrometeorologia da EMBRAPA, Dr. Gustavo Trentin e análise dos dados disponíveis para uso no trabalho. A partir do referencial produzido foi definido que o módulo trataria das variáveis temperatura, precipitação, umidade do ar e insolação por serem as principais variáveis para as questões de interesse na agropecuária.

Levantamento do referencial teórico A terceira etapa, realizada em paralelo à segunda e à quarta, produziu o levantamento do referencial teórico necessário para subsídio da realização deste trabalho. A metodologia usada foi a de revisão sistemática da literatura: inicialmente foram definidas palavras e termos chaves para a pesquisa bibliográfica, juntamente com fontes de dados referentes a publicações científicas.

Figura 2 – Diagrama de sequência da metodologia



Fonte: Autor (2019)

As palavras e termos pesquisados estão listados abaixo:

- agricultura familiar;
- agricultura no Rio Grande do Sul;
- agropecuária;
- clima do Rio Grande do Sul;
- climatologia;
- C#;
- distribuição de probabilidades;
- El Niño e La Niña;
- El Niño Oscilação Sul;
- geradores de valores aleatórios;
- simulação;
- jogos sérios;
- meteorologia;
- modelagem;
- R;

- simulação climática;
- Unity.

Após a definição das palavras, foram definidas bases de dados onde seriam realizadas as pesquisas, as palavras foram pesquisadas em português e em língua inglesa, nas bases de dados que se fez necessário. As bases de dados são listadas abaixo:

- Biblioteca da ACM (https://dl.acm.org/);
- IEEE Xplorer Digital Library (https://ieeexplore.ieee.org/Xplore/home.jsp);
- International Journal of Games Technology (https://www.hindawi.com/journals/ijcgt/);
- Anais do Simpósio Brasileiro de Games e Entretenimento Digital – SBGames (https://www.sbgames.org/);
- SciELO (https://scielo.org/).

Também foram utilizadas as páginas dos mecanismos de busca Google e Google Scholar, livros disponíveis na biblioteca do Campus Bagé da Universidade Federal do Pampa e materiais disponibilizados nas bases de dados das seguintes instituições: EMBRAPA, FEE, IBGE, INMET e INPE. Trabalhos descrevendo aspectos meteorológicos, climatológicos e sua relação com a agropecuária foram utilizados independente do ano, desde que contivessem material de interesse e que fossem provenientes de fontes confiáveis.

O referencial teórico é apresentado no capítulo 2.

Escolha da base de dados climáticos A quarta etapa consistiu na definição do banco de dados a ser utilizado nesta pesquisa. Para este trabalho foi necessário um banco de dados que contenha séries históricas referentes a dados climáticos no Brasil, os mesmos foram analisados em busca de distribuições de probabilidade que pudessem os representar. Optou-se por utilizar o BDMEP (Banco de Dados Meteorológicos para Ensino e Pesquisa) pois ele abriga dados meteorológicos diários de séries históricas das estações meteorológicas da rede de estações do INMET, com diversas informações referentes às medições diárias, de acordo com as normas técnicas internacionais da Organização Meteorológica Mundial. No BDMEP estão acessíveis os dados diários a partir de 1961 das estações para as quais se disponha, em forma digital, de pelo menos 80% dos dados que foram registrados naquele período. Os dados históricos referentes a períodos anteriores a 1961 não

estão em forma digital e, portanto, estão indisponíveis no BDMEP. As séries históricas das variáveis atmosféricas disponíveis para consultas no BDMEP são disponibilizadas de três maneiras: dados horários, dados diários e dados mensais. Os dados disponibilizados de forma horária são referentes à 00h, 12h e 18h, todos os dados com suas unidades são listados a seguir:

- Temperatura de bulbo seco; (°C)[1]
- Temperatura de bulbo úmido (°C);
- Umidade relativa (%);
- Nebulosidade (Código);
- Pressão atmosférica no nível da estação(mbar);
- Velocidade do vento (m/s);
- Direção do vento (Código).

Os dados disponibilizados de forma diária são listados a seguir:

- Precipitação acumulada no dia (mm);
- Temperatura máxima (°C);
- Temperatura mínima (°C);
- Insolação (horas);
- Evaporação do piche (mm);
- Umidade relativa média (%);
- Temperatura compensada média (°C);
- Velocidade média do vento (m/s).

Os dados disponibilizados de forma mensal são listados a seguir:

- Precipitação acumulada no mês (mm);
- Número de dias com precipitação;
- Temperatura máxima média (°C);
- Temperatura mínima média (°C);
- Insolação total (horas);
- Evaporação do piche mensal (mm);

[1]Termômetro de bulbo seco e termômetro de bulbo úmido são componentes de um psicrômetro, aparelho que contém os dois termômetros idênticos colocados um ao lado do outro. A diferença entre esses termômetros é que um deles trabalha com o bulbo seco e o outro tem seu bulbo envolvido por um material umedecido, usualmente algum tecido ou algodão. Esse aparelho é utilizado para a determinação do ponto de orvalho e da umidade relativa do ar.

- Umidade relativa média mensal (%);
- Temperatura compensada média (^{o}C);
- Velocidade média do vento (m/s);
- Velocidade máxima do Vento Média (m/s);
- Direção do vento predominante (código)
- Evapotranspiração potencial (mm);
- Evapotranspiração Real (mm);
- Nebulosidade média (código);
- Visibilidade média (%).

Entre os dados listados, foram utilizados neste trabalho os dados referentes a variações de temperaturas, acúmulos de precipitação, insolação e umidade relativa do ar, conforme descrito na segunda etapa desta metodologia.

Estudo e análise dos dados obtidos A quinta etapa consistiu em analisar os dados obtidos no BDMEP com o objetivo de determinar distribuições de probabilidades capazes de representá-los de maneira factível. As análises foram feitas com a linguagem de programação R, linguagem orientada a objetos com fluxo de execução funcional, que possui ambiente de desenvolvimento integrado. A existência de uma grande quantidade de pacotes para cálculos estatísticos e produção de gráficos de alta qualidade (MATLOFF, 2011), fez com que fosse escolhida para o desenvolvimento dos programas e das análises de dados realizadas neste trabalho. Com o propósito de determinar a melhor distribuição estatística para representar as séries históricas dos dados da cidade de Bagé (RS) foram produzidos testes estatísticos, descritos no Capítulo 4. A análise dos resultados obtidos com esses testes permitiu a definição de quais distribuições seriam implementadas os geradores de valores aleatórios.

Implementação do módulo A sexta etapa consistiu em implementar o módulo sobre o motor (*engine*) de desenvolvimento de jogos *Unity* dentro dos princípios do projeto e desenvolvimento orientados a objetos (MENARD, 2011), utilizando a linguagem padrão da *Unity*, C#. Dessa maneira, poder-se-á substituir o módulo por qualquer outro módulo meteorológico construído sobre dados de outras regiões, tornando a ferramenta flexível tanto para os produtores (que necessitam de dados meteorológicos de sua região) como para pesquisadores que desejem explorar condições meteorológicas como parte de suas pesquisas.

Inicialmente foi implementado o gerador de valores aleatórios para as distribuições definidas na etapa anterior. Após, foram implementados os métodos gerenciadores do clima dentro do SIMCOW.

Testes e validação Foram realizados testes de integração ao jogo SIMCOW e verificação se os dados gerados durante a execução eram consistentes com dados esperados para cada mês. A validação deveria ter sido realizada juntamente aos pesquisadores da EMBRAPA Pecuária Sul, especialistas em Agrometeorologia. Contudo, ainda não houve tempo hábil para realizar essa verificação. Considera-se, contudo, que a construção do sistema sobre dados de realidade provê evidências de que o sistema pode ser usado corretamente, dentro de uma margem de erro aceitável.

As Figuras 3 e 4 apresentam o cronograma das tarefas dividas entre TCC I e TCC II.

Figura 3 – Tarefas do TCC I

Nome	Data inicial	Data final
Levantamento do referencial teórico	13/08/18	11/11/18
Levantamento de variaveis climáticas	13/08/18	01/09/18
Escrever TCCI	01/09/18	22/11/18
ìnicio do estudo e análise dos dados	01/11/18	20/11/18
Preparação para apresentar TCCI	23/11/18	07/12/18
Apresentacão TCCI	08/12/18	08/12/18

Fonte: Autor (2019)

Figura 4 – Tarefas do TCC II

Nome	Data inicial	Data final
Escrever TCCII	05/08/19	10/11/19
Estudo e análise dos dados	05/08/19	30/09/19
Implementação do módulo	30/09/19	31/10/19
Testes e validação	01/10/19	31/10/19
Entrega da versão rascunho	01/11/19	01/11/19
Revisão de texto	02/11/19	10/11/19
Entrega da versão final	10/11/19	10/11/19
Apresentação TCCII	26/11/19	26/11/19

Fonte: Autor (2019)

4 DESENVOLVIMENTO

4.1 Obtenção dos dados

O ambiente de desenvolvimento usado para programação na linguagem R foi o *software* RStudio. O pacote `inmetr` (TATSCH, 2017) foi usado com o intuito de agilizar a obtenção de dados do BDMEP. Esse pacote permite o download da base diretamente para o diretório do R, com ajuste automático dos tipos de dados e preenchimento dos dados incompletos com NA, o que indica ao R que se trata de dados ausentes. Os seguintes comandos são realizados para instalar o inmetr no R:

```
LIBRARY(DEVTOOLS)
INSTALL_GITHUB('LHMET/INMETR')
```

A Figura 5 apresenta os comandos necessários para realizar o download dos dados. O identificador (id) da estação meteorológica de Bagé é 83980, dado que pode ser obtido no site do INMET ou no próprio `inmetr` com o comando `BDMEP_META`, que lista todas as estações e os seus respectivos dados: id, altitude, longitude e altura em relação ao nível do mar.

Figura 5 – Comandos necessários para realizar o download de dados do BDMEP

```
start_date <- "01/01/1961"                                  # Inicialização data Inicial
end_date <- "31/12/2018"                                    # Inicialização data Final
met_data <- bdmep_import(id = 83980,                        # id da Estação
                         sdate = start_date,
                         edate = end_date,
                         email = "eduardocarvalhot@gmail.com",   # e-mail cadastrado no inmet
                         passwd = "senha",                  # senha cadastrada no inmet
                         verbose = TRUE)
```

Fonte: Autor (2019)

Após o carregamento dos dados foi possível analisá-los apropriadamente.

4.2 Análise dos dados

Foram analisados os dados da estação do INMET desde o ano de 1961 até 2018, ou seja, todos os dados que foram disponibilizados de forma digital.

O teste de Kolmogorov-Smirnov (D'AGOSTINO, 1986), conforme levantado no referencial teórico, é um dos testes mais utilizados para testar a aderência de dados climáticos a distribuições de probabilidade. O mesmo foi realizado para sete distribuições de probabilidade, com o objetivo de verificar a aderência dos dados a cada uma das

distribuições, sendo elas: Cauchy, Qui-quadrado, Exponencial, Gama, Log Normal, Normal e Weibull. Essas distribuições foram escolhidas empiricamente, por serem as distribuições contínuas mais comuns de dados com componentes aleatórios e possuírem teoria robusta construída sobre elas.

Para a realização do teste de Kolmogorov-Smirnov é necessário conhecer os parâmetros descritores para cada distribuição. Os parâmetros necessários foram calculados com o pacote `MASS`, que possui a função `FITDISTR`. Após a obtenção dos parâmetros, o teste Kolmogorov-Smirnov foi realizado no R utilizando a função `KS.TEST` do pacote `stats`.

A seguir são listadas as distribuições e os comandos utilizados para cada uma:

Cauchy: Obtenção dos parâmetros:

FITDISTR(DADOS,"CAUCHY"): retorna os parâmetros *location* (localização) e *scale* (escala).

Teste de Kolmogorov-Smirnov:

KS.TEST(DADOS,"PCAUCHY", LOCATION, SCALE)

χ^2: Obtenção dos parâmetros:

FITDISTR(DADOS,"CHI-SQUARED"): retorna o parâmetro df(degrau de liberdade).

Teste de Kolmogorov-Smirnov:

KS.TEST(DADOS,"PCHISQ", DF)

Exponencial: Obtenção dos parâmetros:

FITDISTR(DADOS,"EXPONENTIAL"): retorna o parâmetro rate(taxa).

Teste de Kolmogorov-Smirnov:

KS.TEST(DADOS,"PEXP", RATE)

Gama: Obtenção dos parâmetros:

FITDISTR(DADOS,"GAMMA"): retorna os parâmetros rate(taxa) e shape(forma).

Teste de Kolmogorov-Smirnov:

KS.TEST(DADOS,"PGAMMA", SHAPE, RATE)

Log-normal: Obtenção dos parâmetros:

FITDISTR(DADOS,"LOGNORMAL"): retorna os parâmetros meanlog(média logarítmica) e sdlog(desvio padrão logarítmico).

Teste de Kolmogorov-Smirnov:

KS.TEST(DADOS,"PLNORM", MEANLOG, SDLOG)

Normal: Obtenção dos parâmetros:

FITDISTR(DADOS,"NORMAL"): retorna os parâmetros mean(média) e sd(desvio padrão).

Teste de Kolmogorov-Smirnov:

KS.TEST(DADOS,"PNORM",MEAN,SD)

Weinbull: Obtenção dos parâmetros:

FITDISTR(DADOS,"WEIBULL"): retorna os parâmetros shape(forma) e scale(escala).

Teste de Kolmogorov-Smirnov:

KS.TEST(DADOS,"PWEIBULL", SHAPE, SCALE)

O teste de Kolmogorov-Smirnov tem como retorno dois valores: o valor de p e o valor D. Um valor de p pequeno significa que a probabilidade de obter um valor da estatística de teste como o observado é muito improvável, levando assim à rejeição da hipótese em valores de p menores que 0,05 (SPIEGEL; SCHILLER; SRINIVASAN, 2016). O valor da estatística D do teste de aderência de Kolmogorov-Smirnov informa a máxima distância entre as probabilidades empíricas e as teóricas obtidas sob a função de distribuição de probabilidade em teste. Assim, menores valores da estatística D fornecem maiores valores de p e, consequentemente, maior evidência de não-rejeição da hipótese, ou seja, maior aderência dos dados à distribuição em teste.

Os meses mais afetados pelo fenômeno El Niño Oscilação Sul, conforme descrito no Capítulo 2 são os meses de outubro, novembro e dezembro no ano inicial e os meses de abril, maio e junho no ano seguinte. O restante dos meses também sofrem influência, porém, com menor intensidade. Os meses que sofrem maior influência foram divididos em três casos: anos sem a ocorrência do fenômeno, anos com a ocorrência de El Niño e anos com a ocorrência de La Niña. Foram analisados os dados de precipitação acumulada, dias com precipitação no mês, temperatura mínima, temperatura máxima, horas de insolação e umidade.

As Tabelas 2, 3, 4, 5, 6 e 7 são todas referentes ao mês de janeiro e exemplificam os resultados dos testes realizados.

Na tabela 2, referente ao teste realizado para os dados de precipitação acumulada no mês de janeiro, é possível observar que o maior valor de p, assim como o menor valor D foram obtidos para a distribuição Gama, sendo assim, a distribuição Gama, entre as distribuições testadas, foi a que teve melhor aderência aos dados de precipitação testados.

Tabela 2 – Precipitação acumulada no mês de Janeiro

Distribuição	Valor de p	D
Cauchy	0.1962	0.14797
Qui Quadrado	8.463e-07	0.37565
Exponencial	0.0408	0.19162
Gama	0.9151	0.077307
Log-Normal	0.8758	0.081976
Normal	0.2429	0.14231
Weibull	0.8633	0.083311

Fonte: Autor (2019)

Na tabela 3, referente aos testes realizados para os dados de quantidade de dias com precipitação no mês de janeiro, é possível observar que o maior valor de p, assim como o menor valor D foram obtidos para a distribuição Log-Normal, sendo assim, a distribuição Log-normal, entre as distribuições testadas, foi a que teve melhor aderência aos dados de dias com precipitação.

Tabela 3 – Dias com precipitação no mês de Janeiro

Distribuição	Valor de p	D
Cauchy	0.6763	0.14137
Qui Quadrado	0.5089	0.16119
Exponencial	0.00127	0.37626
Gama	0.6152	0.14848
Log-Normal	0.6919	0.13955
Normal	0.2867	0.19313
Weibull	0.3244	0.18681

Fonte: Autor (2019)

A tabela 4, apresenta os resultados referente aos testes realizados para os dados de temperaturas mínimas no mês de janeiro, é possível observar que o maior valor de p,

assim como o menor valor *D* foram obtidos para a distribuição Log-Normal, sendo assim, a distribuição Log-normal, entre as distribuições testadas, foi a que teve melhor aderência aos dados de temperaturas mínimas.

Tabela 4 – Temperatura mínima no mês de Janeiro

Distribuição	Valor de p	D
Cauchy	0.571	0.10768
Qui Quadrado	1.07e-06	0.37138
Exponencial	8.882e-16	0.58589
Gama	0.7766	0.090239
Log-Normal	0.8193	0.086315
Normal	0.691	0.097599
Weibull	0.2547	0.14019

Fonte: Autor (2019)

A tabela 5, apresenta os resultados referente aos testes realizados para os dados de temperaturas máximas no mês de janeiro, é possível observar que o maior valor de *p*, assim como o menor valor *D* foram obtidos para a distribuição Log-Normal, sendo assim, a distribuição Log-normal, entre as distribuições testadas, foi a que teve melhor aderência aos dados de temperaturas máximas.

Tabela 5 – Temperatura máxima no mês de Janeiro

Distribuição	Valor de p	D
Cauchy	0.6595	0.098347
Qui Quadrado	1.916e-07	0.38496
Exponencial	2.2e-16	0.59861
Gama	0.8336	0.08334
Log-Normal	0.8427	0.082453
Normal	0.7824	0.088031
Weibull	0.3934	0.12165

Fonte: Autor (2019)

A tabela 6, apresenta os resultados referente aos testes realizados para os dados de horas com insolação no mês de janeiro, é possível observar que o maior valor de *p*,

assim como o menor valor *D* foram obtidos para a distribuição Normal, sendo assim, a distribuição Normal, entre as distribuições testadas, foi a que teve melhor aderência aos dados de horas com insolação.

Tabela 6 – Horas de insolação no mês de Janeiro

Distribuição	Valor de p	D
Cauchy	0.7057	0.098486
Qui Quadrado	0.1689	0.15565
Exponencial	3.601e-12	0.5149
Gama	0.9855	0.063842
Log-Normal	0.9801	0.065771
Normal	0.9964	0.057031
Weibull	0.9835	0.064596

Fonte: Autor (2019)

A tabela 7, apresenta os resultados referente ao testes realizados para os dados de umidade relativa no mês de janeiro, é possível observar que o maior valor de *p*, assim como o menor valor *D* foram obtidos para a distribuição de Weibull, sendo assim, a distribuição de Weibull, entre as distribuições testadas, foi a que teve melhor aderência aos dados de horas com insolação.

Tabela 7 – Umidade relativa no mês de Janeiro

Distribuição	Valor de p	D
Cauchy	0.2679	0.14416
Qui Quadrado	0.04	0.20215
Exponencial	2.401e-12	0.52516
Gama	0.3734	0.13129
Log-Normal	0.3448	0.13451
Normal	0.4963	0.11892
Weibull	0.955	0.072417

Fonte: Autor (2019)

As tabelas para os demais meses que sofrem pouca influência do fenômeno ENOS encontram-se no Apêndice B. A Tabela 8 apresenta o resultado da análise dos meses com pouca influência do fenômeno. Nela é relacionado a variável, o mês e a distribuição que teve melhor aderência para cada caso.

Tabela 8 – Meses com baixa influência do fenômeno ENOS

Mês	Precipitação	Dias com precipitação	Temperatura Mínima	Temperatura Máxima	Horas de Insolação	Umidade
Janeiro	Gama	Log-Normal	Log-Normal	Log-Normal	Normal	Weibull
Fevereiro	Weibull	Weibull	Cauchy	Log-Normal	Weibull	Weibull
Março	Log-Normal	Log-Normal	Log-Normal	Cauchy	Normal	Cauchy
Julho	Weibull	Gama	Weibull	Gama	Weibull	Normal
Agosto	Weibull	Gama	Gama	Log-Normal	Weibull	Normal
Setembro	Gama	Log-Normal	Gama	Cauchy	Weibull	Weibull

Fonte: Autor (2019)

Optou-se por não criar tabelas de todos os testes dos meses com influência do ENOS, devido ao fato de que seriam criadas 18 tabelas para cada mês, resultando em um total de 108 tabelas.

Os valores de p e D obtidos nesses testes foram testados no R, através dos comandos MIN(VETORVALORESD) e MAX(VETORVALORESP) afim de identificar o maior valor de p e o menor valor D obtidos. A identificação da distribuição se fez pelos comandos WHICH.MIN(VETORVALORESD) e WHICH.MAX(VETORVALORESP) que retornam a posição do vetor no qual está contido o menor valor D e o maior valor de p. Sendo que as posições nos vetores foram organizadas da seguinte forma: posição 1 contém os resultados para o teste de aderência a distribuição de Cauchy, posições seguintes seguem em ordem alfabética até a posição 7, referente aos resultados para a distribuição de Weibull.

Com base nesses valores, as relações de meses, variáveis e distribuições com melhor aderência foram passados diretamente para a tabela 9 que apresenta as distribuições que obtiveram melhor aderência nas três variações existentes para cada mês.

4.3 Implementação do gerador de valores aleatórios

A análise dos dados realizada definiu que deveriam ser implementados geradores de valores aleatórios para as seguintes distribuições: Cauchy, Gama, Log-Normal, Normal e Weibull, visto serem as distribuições que representam os dados analisados. Também foi necessário implementar o gerador para a distribuição Uniforme, visto que o mesmo foi utilizado na implementação das outras distribuições.

Para a implementação do gerador de valores aleatórios na *Unity*, além do referencial teórico sobre a geração de valores aleatórios, foram analisadas implementações existentes das distribuições definidas. Incluindo versões em C (MARSAGLIA; TSANG, 2000), C++ (COOK, 2014b) e Java (ZAFALON; JR, 2006) .

Para implementar o gerador de valores aleatórios da distribuição Uniforme foi utilizado algoritmo *Multiply-With-Carry* (MARSAGLIA; ZAMAN, 1991), descrito no Capítulo 2, que funciona como base para a implementação dos geradores das demais distribuições que foram implementadas. O algoritmo implementado segue os seguintes passos:

1. Definir dois valores inteiros u e v (gerados em função do horário do sistema)
2. Fazer $v \leftarrow 36969 * (v \text{ AND } 65535) + (v \text{ DESLOCADO 16 bits a direita})$
3. Fazer $u \leftarrow 18000 * (u \text{ AND } 65535) + (u \text{ DESLOCADO 16 bits a direita})$

Tabela 9 – Meses com influência do fenômeno ENOS

Mês	Precipitação	Dias com precipitação	Temperatura Mínima	Temperatura Máxima	Horas de insolação	Umidade Relativa
Abril sem ENOS	Weibull	Normal	Log-Normal	Cauchy	Cauchy	Weibull
Abril com El Niño	Weibull	Cauchy	Cauchy	Cauchy	Weibull	Weibull
Abril com La Niña	Cauchy	Cauchy	Cauchy	Normal	Weibull	Gama
Maio sem ENOS	Normal	Normal	Gama	Normal	Weibull	Normal
Maio com El Niño	Normal	Cauchy	Cauchy	Cauchy	Cauchy	Weibull
Maio com La Niña	Cauchy	Log-Normal	Cauchy	Gama	Cauchy	Normal
Junho sem ENOS	Weibull	Log-Normal	Cauchy	Normal	Cauchy	Weibull
Junho com El Niño	Cauchy	Normal	Weibull	Gama	Normal	Gama
Junho com La Niña	Cauchy	Log-Normal	Cauchy	Cauchy	Cauchy	Weibull
Outubro sem ENOS	Normal	Gama	Weibull	Gama	Gama	Weibull
Outubro com El Niño	Log-Normal	Gama	Weibull	Cauchy	Weibull	Gama
Outubro com La Niña	Cauchy	Cauchy	Normal	Normal	Weibull	Normal
Novembro sem ENOS	Weibull	Gama	Normal	Weibull	Normal	Normal
Novembro com El Niño	Cauchy	Cauchy	Normal	Cauchy	Normal	Cauchy
Novembro com La Niña	Gama	Cauchy	Weibull	Log-Normal	Weibull	Normal
Dezembro sem ENOS	Weibull	Cauchy	Gama	Log-Normal	Log-Normal	Normal
Dezembro com El Niño	Log-Normal	Cauchy	Gama	Log-Normal	Gama	Cauchy
Dezembro com La Niña	Cauchy	Cauchy	Cauchy	Cauchy	Log-Normal	Weibull

Fonte: Autor (2019)

4. Fazer $x \leftarrow (v \text{ DESLOCADO 16 bits a direita}) + u$
5. Retornar $(x+1) * \frac{1}{2^{32}+2}$

O resultado retornado pela função descrita acima sempre estará no intervalo $[0,1]$.

O gerador para a distribuição Normal foi implementado utilizando-se da transformação de Box-Müller, conforme descrito no Capitulo 2, o método é utilizado para gerar duas distribuições Normal-padrão independentes, dado um conjunto de dados aleatórios uniformemente distribuídos.

1. Gerar distribuições Uniforme: U_1, U_2.
2. Fazer $R \leftarrow \sqrt{-2ln(U_1)}$.
3. Fazer $\theta \leftarrow 2\pi U_2$.
4. Fazer $X_1 \leftarrow R\cos(\theta)$.
5. Fazer $X_2 \leftarrow R(\theta)$.
6. Retornar X_1 e X_2.

Em (MARSAGLIA; TSANG, 2000) é descrito um algoritmo para geração da distribuição Gama tendo *shape*(forma) como parâmetro de entrada e sua implementação é exemplificada utilizando a linguagem C. O algorítimo tem a seguinte sequência de operações:

1. Gerar distribuição Normal X_1 com média 1 e desvio padrão 0.
2. Fazer $d \leftarrow \ shape \ - \frac{1}{3}, c \leftarrow \frac{1}{\sqrt{(9d)}}$.
3. Fazer $v \leftarrow (1 + c * X_1)^3$.
4. Repetir itens 2 e 3 se $v \leq 0$
5. Gerar distribuição Uniforme U_1.
6. Verificar se $U_1 < 1 - 0,0331 * X_1^4$
7. Verificar se $v > 0$ e $\log(U_1) < (0,5 * X_1^2) - d(d * v) + (d * \log(v))$
8. Retornar $d * v$

Segundo (COOK, 2014a) tendo como parâmetros de entrada $\beta = scale$ e $\gamma = shape$ o seguinte algorítimo é a forma mais simples de se obter a Distribuição de Weibull:

1. Gerar Distribuição Uniforme U_1
2. Retornar $\beta * (-\log(U_1)^{\frac{1}{V}})$

Ainda segundo (COOK, 2014a), tendo como parâmetros de entrada a média e a escala a distribuição de Cauchy é gerada da seguinte forma:

1. Gerar Distribuição Uniforme U_1
2. Retornar $Media + (Escala * \tan(\pi * (U_1 - 0,5)))$

A distribuição Log-normal é obtida da seguinte maneira (COOK, 2014a):

1. Gerar Distribuição Normal X_1
2. Retornar e^{X_1}

4.4 Implementação do módulo climático

Duas novas classes foram adicionadas ao projeto original do SIMCOW, sendo elas:

- GeradorValores: Classe que contêm a implementação do gerador de valores aleatórios;
- ModuloClimatico: Classe que contêm a implementação dos cálculos relacionados as variáveis climáticas. A mesma substituiu a classe da implementação original.

A proposta de implementação dos módulos externos apresentada por (ROBAINA, 2018) previa que para a adição de módulos externos nenhum código da arquitetura original seria modificado, o que se provou equivocado, já que foi necessário modificar as seguintes classes:

- Clima: Classe que contêm variáveis descritoras do clima, faz uso do módulo climático para definição de seus valores. Possuía variáveis referentes a precipitação momentânea, temperatura momentânea e horas com luminosidade no dia. Foram adicionadas variáveis referentes aos seguintes dados : temperaturas mínima e máxima do dia, precipitação acumulada diária, precipitação acumulada mensal, quantidade de dias com precipitação no mês, horas de insolação mensal e umidade relativa.
- GM: *Game Manager*, classe responsável por gerenciar todo o funcionamento do jogo. Foi modificada a forma como ela gerencia o clima, anteriormente ao inicio de um novo dia eram atualizados todos os dados climáticos, que ficavam estáticos até um novo dia iniciar, a funcionalidade de atualização diária foi mantida e é utilizada para atualizar os dados de qual será a precipitação total no dia, quais serão as temperaturas mínima e máxima, assim como, qual será a insolação diária e umidade relativa .

Foi adicionada a funcionalidade de atualização do clima ao inicio de um novo mês, utilizada para calcular dados como qual será a precipitação total no mês, quantos dias ocorrerão precipitação e qual será a insolação total ao final do mês.
Também foi adicionada a funcionalidade de atualização do clima a cada hora, responsável por definir os dados de temperatura, precipitação, umidade e insolação momentâneos.

O cálculo da precipitação no módulo climático segue os seguintes passos:

1. Gerar com a distribuição adequada para o mês que se iniciou qual será a precipitação total ao longo do mesmo;
2. Gerar com a distribuição adequada para o mês que se iniciou a quantidade de dias que ocorrerão precipitações;
3. Calculo da precipitação diária é feito dividindo-se a precipitação mensal pela quantidade de dias com chuva restantes;
4. Definição de que se haverá precipitação no dia é feita através de um *randon* booleano, respeitando o limite de precipitação e dias com precipitação calculados anteriormente. Desta forma a precipitação não se concentrara toda no inicio do mês, ocorrendo dias com e sem chuva ao longo do mesmo;
5. Se ocorrer precipitação no dia, a definição de que ocorrera precipitação horaria também é feita através de um *randon* booleano. Desta forma a precipitação não estará toda concentrada no incio do dia, ocorrendo horas com e sem chuva ao longo do mesmo;
6. A precipitação horária é calculada dividindo-se a precipitação diária calculada anteriormente por 10 e após é feito um *randon* de 1 até o valor calculado. Desta forma a precipitação diária não ocorrera toda no inicio do dia e serão gerados diferentes valores de precipitações momentâneas ao longo dia.

O calculo da variação de temperatura no módulo climático segue os seguintes passos:

1. Gerar com a distribuição adequada a temperatura mínima para o dia;
2. Gerar com a distribuição adequada a temperatura máxima para o dia;
3. É definida uma variável chamada variador. Esta variável recebe a diferença entre a temperatura mínima e a máxima calculada e o resultado é divido por 24;
4. Ao inicio do novo dia, a temperatura é definida como a temperatura mínima

incrementada em cinco vezes o variador;

5. A cada hora a temperatura é decrementada em uma vez o variador, resultando na ocorrência da temperatura mínima as 5h da manhã;
6. Nas horas seguintes a temperatura é incrementada em três vezes o variador, resultando na temperatura máxima as 14h;
7. Após atingir a temperatura máxima, a temperatura começa a ser decrementada em duas vezes o variador, isso resultara na temperatura ao final do dia sendo a temperatura mínima mais quatro vezes o variador;

O calculo da insolação no módulo climático segue os seguintes passos:

1. Gerar com a distribuição adequada a insolação mensal
2. Distribuir no mês, considerando que momentos com precipitação são reduzidos da insolação diária.

A umidade relativa é gerada hora a hora utilizando-se da distribuição adequada para o mês em questão.

A Figura 6 demonstra a tela de gerência do jogo SIMCOW mostrando os dados climáticos que estão sendo calculados durante a execução: temperaturas mínima e máxima para o dia, temperatura atual, luminosidade diária, luminosidade mensal, umidade relativa, dias com precipitação no mês, precipitação total para o mês, precipitação acumulada até o presente momento, precipitação diária e precipitação momentânea. Importante salientar que dados que representam o futuro do jogo, como qual será a precipitação e a luminosidade ao final do mês não serão exibidos na tela da versão final para o jogador.

Figura 6 – Tela de gerência do jogo SIMCOW

Fonte: Autor (2019)

4.5 Testes

Foram realizados testes de regressão nas classes alteradas durante o desenvolvimento, durante os testes não foram detectados problemas em funções que já existiam na implementação inicial do SIMCOW (ROBAINA, 2018), sendo assim, as novas funcionalidades não comprometeram as já existentes.

Durante a implementação do gerador de valores aleatórios foram realizados testes unitários das funções implementadas afim de validação dos dados gerados, o mesmo procedimento foi adotado para as funções do módulo climático.

5 CONSIDERAÇÕES FINAIS

O objetivo deste trabalho, conforme apresentado no Capítulo 1, era construir um módulo climático, baseado em dados da realidade, para poder ser usado no jogo sério SIMCOW (ROBAINA; TEIXEIRA; FERREIRA, 2017; ROBAINA; FERREIRA, 2018; ROBAINA, 2018) como um modelo realístico para o propósito de simulação de operações de manejo em propriedades rurais. Para tanto, foram levantadas variáveis meteorológicas relevantes para os processos agropecuários e buscados dados na base do INMET relacionados ao dados da cidade de Bagé, RS.

A análise dos dados obtidos foi realizada por meio de testes estatísticos de *model fitting*, com o intuito de descobrir quais as distribuições teóricas que melhor representavam os dados de cada uma das variáveis, levando em consideração os fenômenos climáticos relevantes, com ênfase nos fenômenos El Niño e La Niña. Após essa análise, foram desenvolvidos algoritmos de geração de números aleatórios dentro das distribuições pertinentes, para modelar os dados diários do tempo. Os algoritmos desenvolvimentos foram incorporados ao jogo, na plataforma de desenvolvimento Unity.

O trabalho desenvolvido resultou em um módulo climático mais aproximado da realidade quando comparado ao módulo original do jogo. O módulo original apresentava de forma estática os dados diários referentes ao ano de 2017. A Tabela 10 apresenta uma comparação entre as duas implementações.

Quanto à fidedignidade do módulo implementado com a realidade, pode-se dizer que os valores gerados, como a precipitação mensal e as temperaturas mínimas e máximas para os dias, aproximam-se dos valores reais, porém, a forma como eles foram distribuídos são aproximações não tão fiéis à realidade. Grandes variações de temperatura em um curto período de horas, assim como grandes volumes de chuvas em poucos dias ou até mesmo em único dia, situações que ocorrem no mundo real, não acontecerão na versão atual do módulo. Essas particularidades do clima no RS (ou em outro lugar) devem ser trabalhadas construindo distribuições que permitam modelar a variação horária de temperatura, por exemplo. Outro fator a ser considerado é que os dados de insolação obtidos no BDMEP trazem a quantia de horas com luz solar que, apesar de ser um dado importante, seria melhor representado em conjunto com dados de radiação solar, já que a mesma influencia diretamente no crescimento da vegetação. Dessa forma, para que um novo módulo mais completo seja construído, será necessário o uso de outros dados além daqueles disponibilizados pelo INMET.

Tabela 10 – Comparação entre os módulos climáticos

Variável	Módulo original	Módulo desenvolvido
Precipitação	Chove 10mm por dia até alcançar a precipitação do mês em 2017. Toda a chuva concentrada nos primeiros dias do mês.	Precipitação e quantia de dias com precipitação são calculados no inicio do mês. Dias com chuva são distribuídos de forma randômica ao longo do mês. Horários com chuva e precipitação momentânea também são variáveis.
Temperatura	Com base na temperatura mínima média e temperatura máxima média do mês em 2017, o módulo calcula uma temperatura intermediaria no inicio de um novo dia. O dia permanece com a mesma temperatura durante o seu decorrer.	Temperatura mínima e máxima são calculadas ao inicio de um novo dia. A temperatura mínima ocorre no período da madrugada e a máxima no período da tarde.
Horas de insolação	Horas de insolação mensais do ano de 2017 dividas igualmente entre os dias.	Horas calculadas ao inicio do mês e divididas considerando que dias com precipitação no período diurno terão menos horas de insolação.
Umidade	O módulo não possui dados relacionados a umidade.	Umidade calculada e atualizada hora a hora.

Fonte: Autor (2019)

REFERÊNCIAS

ALTIOK, T.; MELAMED, B. **Simulation modeling and analysis with Arena**. [S.l.]: Elsevier, 2010.

BACK, A. Seleção de distribuição de probabilidade para chuvas diárias extremas do estado de santa catarina. **Revista Brasileira de Meteorologia**, v. 16, n. 2, p. 211–222, 2001.

BOX, G. E. P.; MULLER, M. E. A note on the generation of random normal deviates. **Ann. Math. Statist.**, v. 29, n. 2, p. 610–611, 1958.

CAMPOS, H. d. **Estatística experimental não-paramétrica**. Piracicaba: ESALQ, 1983. 349 p.

CERA, J. C.; FERRAZ, S. E. T. Variações climáticas na precipitação no sul do brasil no clima presente e futuro. **Revista Brasileira de Meteorologia**, SciELO Brasil, v. 30, n. 1, 2015.

COOK, J. D. **Diagram of distribution relationships**. 2014. Disponível em: https://www.johndcook.com/blog/distribution_chart/. Acesso em: 15 out. 2019.

COOK, J. D. **Random number generation using C++ TR1**. 2014. Disponível em: https://www.johndcook.com/blog/cpp_TR1_random/. Acesso em: 2 out. 2019.

CUNHA, G. R. da et al. El niño/la niña-oscilação sul e seus impactos na agricultura brasileira: fatos, especulações e aplicações. **Revista Plantio Direto**, Aldeia Norte Editora, v. 20, n. n. 121, p. p. 18–22, 2011.

D'AGOSTINO, R. **Goodness-of-fit-techniques**. New York: Routledge, 1986.

D'IPOLITTO, C. Jogos de negócio e educação empreendedora. **Sistemas & Gestão**, Rio de Janeiro, n. 7, p. 192–204, 2012.

FERREIRA, D. F. **Estatıstica Computacional Utilizando R**. 2007.

FILHO, A. C.; MATZENAUER, R.; TRINDADE, J. K. da. Ajustes de funções de distribuição de probabilidade à radiação solar global no estado do rio grande do sul. **Pesquisa Agropecuária Brasileira**, SciELO Brasil, v. 39, n. 12, p. 1157–1166, 2004.

FISHER, R. A. The design of experiments. Oliver & Boyd, 1935.

FONTANA, D. C.; BERLATO, M. A. Influência do el niño oscilação sul sobre a precipitação pluvial no estado do rio grande do sul. **Revista Brasileira de Agrometeorologia, Santa Maria**, v. 5, n. 1, p. 127–132, 1997.

FREITAS, C. de. **Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico - 2ª Edição**. Novo Hamburgo: Editora Feevale, 2013. ISBN 9788577171583.

GONÇALVES, F. N.; BACK, Á. J. Análise da variação espacial e sazonal e de tendências na precipitação da região sul do brasil. **Revista de Ciências Agrárias**, Sociedade de Ciências Agrárias de Portugal, v. 41, n. 3, p. 11–20, 2018.

GRANDO, M. Z. Um retrato da agricultura familiar gaúcha. **Textos para Discussão FEE Nº 98**, 2011.

JAMONNAK, S.; CHENG, E. Little botany: A mobile game utilizing data integration to enhance plant science education. In: **International Journal of Computer Games Technology**. [S.l.: s.n.], 2017.

KAYANO, M. et al. El niño e la niña dos últimos 30 anos: diferentes tipos. 11 2016.

KENDALL, M. G. et al. The advanced theory of statistics. **The advanced theory of statistics.**, Charles Griffin and Co., Ltd., London, n. 2nd Ed, 1946.

LAW, A. M. **Simulation modeling and analysis**. Nova Iorque: McGraw-Hill, 2007.

LOULA, A. C. et al. Modelagem ambiental em um jogo eletrônico educativo. In: **VIII Brazilian Symposium on Games and Digital Entertainment. Rio de Janeiro: SBGames**. [S.l.: s.n.], 2009. v. 1, p. 171–180.

LYRA, G. B. et al. Regiões homogêneas e funções de distribuição de probabilidade da precipitação pluvial no estado de táchira, venezuela. **Pesquisa Agropecuária Brasileira**, SciELO Brasil, v. 41, n. 2, p. 205–215, 2006.

MANN, P. S. **Introdução à estatística**. 8st. ed. [S.l.]: LTC, 2008. ISBN 9788521627647.

MARSAGLIA, G. The marsaglia random number cdrom with the diehard battery of tests of randomness. **Distributed by the author (geo@ stat. fsu. edu) from Florida State University**, 1996.

MARSAGLIA, G.; TSANG, W. W. A simple method for generating gamma variables. **ACM Transactions on Mathematical Software (TOMS)**, ACM, v. 26, n. 3, p. 363–372, 2000.

MARSAGLIA, G.; ZAMAN, A. A new class of random number generators. **The Annals of Applied Probability**, JSTOR, p. 462–480, 1991.

MASTROCOLA, V. M. **Ludificador**. São Paulo: Independente, 2012.

MATLOFF, N. S. **Art of R programming**. San Francisco, Calif.: No Starch Press, 2011.

MEESTER, R. **A natural introduction to probability theory**. [S.l.]: Springer Science & Business Media, 2008.

MENARD, M. **Game Development with Unity**. 1st. ed. Boston, MA, United States: Course Technology Press, 2011. ISBN 9781435456587.

MICHAEL, D. R.; CHEN, S. L. **Serious games: Games that educate, train, and inform**. [S.l.]: Muska & Lipman/Premier-Trade, 2005.

NASCIMENTO, W. F. et al. Efeitos da temperatura sobre a soja e milho no estado de mato grosso do sul. **Investigación Agraria**, Dirección de Investigación; Facultad de Ciencias Agrarias, v. 20, n. 1, p. 30–37, 2018.

OES, L. C. Agricultura familiar predomina no brasil. **ComCiência: revista eletrônica de jornalismo científico**, 2011.

PARAVISI, M.; AMORY, A. d. M. Simulating rescue of agents in crowds during emergency situations. In: **SBGames**. [S.l.: s.n.], 2017.

PELETTI, R. B. Micro dentista: Um jogo digital aplicado a saúde bucal. In: **SBGames**. [S.l.: s.n.], 2015.

PHILANDER, S. El niño and la niña. **Journal of the Atmospheric Sciences**, v. 42, n. 23, p. 2652–2662, 1985.

PRIMACK, R. B.; RODRIGUES, E. **Biologia da conservação**. [S.l.: s.n.], 2006.

RAZALI, N. M.; WAH, Y. B. et al. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. **Journal of statistical modeling and analytics**, v. 2, n. 1, p. 21–33, 2011.

REIS, A. V. D.; ROCHA, R. Simuladores para ensino de ressuscitação cardiorrespiratória: Análise do cardiosim como uma solução brasileira. In: **SBGames**. [S.l.: s.n.], 2017.

ROBAINA, R. P. **SIMCOW: JOGO PARA APRENDIZADO DE GESTÃO DE PROPRIEDADES RURAIS**. 200 p. Monografia (Trabalho de Conclusão de Curso) — Universidade Federal do Pampa, Bagé, 2018.

ROBAINA, R. P.; FERREIRA, A. P. L. Projeto de arquitetura modular para jogo sério voltado À gestão de propriedades rurais. In: **10º Salão Internacional de Ensino, Pesquisa e Extensão**. [S.l.: s.n.], 2018.

ROBAINA, R. P.; TEIXEIRA, E. C.; FERREIRA, A. P. L. Aprendizado em gestão de propriedades agrícolas por meio de jogos educacionais. In: **9º Salão Internacional de Ensino, Pesquisa e Extensão**. [S.l.: s.n.], 2017.

ROSA, C. A. **NÚMEROS ALEATÓRIOS: GERAÇÃO,QUALIDADE E APLICAÇÕES**. Monografia (Dissertação de mestrado) — UNIVERSIDADE FEDERAL DO ABC, 2016.

SANSIGOLO, C. A. Distribuições de extremos de precipitação diária, temperatura máxima e mínima e velocidade do vento em piracicaba, sp (1917-2006). **Revista Brasileira de Meteorologia**, SciELO Brasil, v. 23, n. 3, p. 341–346, 2008.

SCHUTZ, D. C. Comparação entre algoritmos geradores das distribuições normal, qui-quadrado, f de snedecor et de studant através de simulação. 2012.

SILVA, J. C. da et al. Análise de distribuição de chuva para santa maria, rs. **Revista Brasileira de Engenharia Agrícola e Ambiental**, SciELO Brasil, v. 11, n. 1, p. 67–72, 2007.

SILVA, L. D. L. D.; ALVES, A. G. Ocean simulator: objeto de aprendizagem sobre a vida marinha. In: **SBGames**. [S.l.: s.n.], 2017.

SILVA, M. A. V. Meteorologia e climatologia. 2006.

SOUZA, A. et al. Calangos: O desenvolvimento de um jogo educacional para o ensino de ecologia e evoluçao. **Anais da X Escola Regional de Computação Bahia-Alagoas-Sergipe**, 2010.

SOUZA, G. S.; JR, N. A. Geradores de números aleatórios. **NOTAS TÉCNICAS**, v. 1, n. 2, 2011.

SPIEGEL, M. R.; SCHILLER, J. J.; SRINIVASAN, R. A. **Probabilidade e Estatística-: Coleção Schaum**. [S.l.]: Bookman Editora, 2016.

TATSCH, J. **inmetr: Historical Data from Brazilian Meteorological Stations in R**. [S.l.], 2017. R package version 0.2.5. Disponível em: https://github.com/lhmet/inmetr.

TEIXEIRA, E. C.; FERREIRA, A. P. L. Projeto de um mÓdulo climatico para um simulador de gestÃo rural. In: **10º Salão Internacional de Ensino, Pesquisa e Extensão**. [S.l.: s.n.], 2018.

WARD, J. H. Hierarchical grouping to optimize an objective function. **Journal of the American Statistical Association**, v. 58, n. 301, p. 236–244, 1963. Disponível em: http://www.tandfonline.com/doi/abs/10.1080/01621459.1963.10500845.

ZAFALON, G. F.; JR, A. M. Construçao de geradores independentes de números aleatórios para diferentes distribuiçoes probabilısticas. In: **Workshop em Computaçao e Aplicaçoes**. [S.l.: s.n.], 2006.

ZEIGLER, B. P.; KIM, T. G.; PRAEHOFER, H. **Theory of modeling and simulation**. [S.l.]: Academic press, 2000.

APÊNDICE A – DOCUMENTO DE REQUISITOS

SIMCOW: Módulo Climático

Documento de Requisitos

Escrito por: Eduardo Carvalho Teixeira

Introdução

Este documento contém a especificação de requisitos para o módulo climático do jogo sério SIMCOW. O objetivo deste módulo é proporcionar mudanças climáticas mais realistas para o jogador. O foco do jogo é o ensino da gestão de propriedades rurais. Portanto, a variação climática será essencial para um bom aproveitamento do jogador. Durante este documento serão utilizadas as seguintes abreviações:

- RF: requisito funcional;
- RNF: requisito não funcional.

Este documento está organizado da seguinte maneira. Inicialmente são apresentados os requisitos funcionais e não funcionais, respectivamente. Para cada requisito é associado um código do requisito, seguido do nome do requisito. Em seguida é indicado a sua prioridade e uma descrição sucinta do requisito.

Requisitos funcionais

[RF - 01] Gerar amostras aleatórias de uma Distribuição Uniforme	
Prioridade: Essencial	O módulo deverá gerar amostras de uma Distribuição Uniforme que serão utilizadas na geração das demais distribuições.

[RF - 02] Gerar amostras aleatórias de uma Distribuição Normal	
Prioridade: Essencial	O módulo deverá gerar amostras de uma Distribuição Normal que serão utilizadas nas variáveis climáticas em que forem aplicáveis.

[RF - 03] Gerar amostras aleatórias de uma Distribuição de Cauchy	
Prioridade: Essencial	O módulo deverá gerar amostras de uma Distribuição de Cauchy que serão utilizadas nas variáveis climáticas em que forem aplicáveis.

[RF - 04] Gerar amostras aleatórias de uma Distribuição Gama	
Prioridade: Essencial	O módulo deverá gerar amostras de uma Distribuição Gama que serão utilizadas nas variáveis climáticas em que forem aplicáveis.

[RF - 05] Gerar amostras aleatórias de uma Distribuição Log-Normal

Prioridade: Essencial	O módulo deverá gerar amostras de uma Distribuição Log-Normal que serão utilizadas nas variáveis climáticas em que forem aplicáveis.

[RF - 06] Gerar amostras aleatórias de uma Distribuição de Weibull

Prioridade: Essencial	O módulo deverá gerar amostras de uma Distribuição de Weibull que serão utilizadas nas variáveis climáticas em que forem aplicáveis.

[RF - 07] Calcular a precipitação total para o mês

Prioridade: Essencial	Ao início de um novo mês, o módulo deverá realizar o cálculo de qual será a precipitação total.

[RF - 08] Calcular os dias com precipitação no mês

Prioridade: Essencial	Ao início de um novo mês, o módulo deverá realizar o cálculo de quantos dias terão precipitação.

[RF – 09] Calcular a precipitação diária e horária

Prioridade: Essencial	Com base nos dados de precipitação total e dias com ocorrência de precipitação no mês, o módulo deverá calcular a precipitação total diária e a horária

[RF – 10] Calcular a temperatura mínima para o dia

Prioridade: Essencial	O módulo deverá calcular a temperatura mínima para o dia.

[RF – 11] Calcular a temperatura máxima para o dia

Prioridade: Essencial	O módulo deverá calcular a temperatura mínima para o dia.

[RF – 12] Variar a temperatura entre a mínima e máxima ao longo do dia

Prioridade: Essencial	O módulo deverá variar a temperatura diária entre a mínima e máxima calculada.

[RF – 13] Calcular a insolação diária

Prioridade: Desejável	O módulo deverá calcular a quantas horas de sol haverá no dia.

[RF – 14] Calcular a umidade relativa

Prioridade: Desejável	O módulo deverá calcular a umidade relativa.

Requisitos não funcionais

[RNF - 01] Ferramenta de Desenvolvimento

Prioridade: Essencial	O módulo deve ser desenvolvido com a *game engine* Unity na versão 2017.02 ou superior.

[RNF - 02] Software

Prioridade: Essencial	Todos os códigos desenvolvidos para o módulo devem ser escritos na linguagem de programação C#. Utilizando as boas práticas de programação previstas no paradigma de orientação à objetos.

[RNF - 03] Documentação

Prioridade: Desejável	Tanto a arquitetura quanto os códigos devem conter uma documentação completa. A documentação deverá seguir os padrões de documentação de software.

[RNF - 04] Realismo na Simulação

Prioridade: Desejável	Os componentes simulados devem possuir comportamento o mais real possível.

APÊNDICE B – TABELAS

B.1 Janeiro

Tabela 11 – Precipitação acumulada no mês de Janeiro

Distribuição	Valor de p	D
Cauchy	0.1962	0.14797
Qui Quadrado	8.463e-07	0.37565
Exponencial	0.0408	0.19162
Gama	0.9151	0.077307
Log-Normal	0.8758	0.081976
Normal	0.2429	0.14231
Weibull	0.8633	0.083311

Fonte: Autor (2019)

Tabela 12 – Dias com precipitação no mês de Janeiro

Distribuição	Valor de p	D
Cauchy	0.6763	0.14137
Qui Quadrado	0.5089	0.16119
Exponencial	0.00127	0.37626
Gama	0.6152	0.14848
Log-Normal	0.6919	0.13955
Normal	0.2867	0.19313
Weibull	0.3244	0.18681

Fonte: Autor (2019)

Tabela 13 – Temperatura mínima no mês de Janeiro

Distribuição	Valor de p	D
Cauchy	0.571	0.10768
Qui Quadrado	1.07e-06	0.37138
Exponencial	8.882e-16	0.58589
Gama	0.7766	0.090239
Log-Normal	0.8193	0.086315
Normal	0.691	0.097599
Weibull	0.2547	0.14019

Fonte: Autor (2019)

Tabela 14 – Temperatura máxima no mês de Janeiro

Distribuição	Valor de p	D
Cauchy	0.6595	0.098347
Qui Quadrado	1.916e-07	0.38496
Exponencial	2.2e-16	0.59861
Gama	0.8336	0.08334
Log-Normal	0.8427	0.082453
Normal	0.7824	0.088031
Weibull	0.3934	0.12165

Fonte: Autor (2019)

Tabela 15 – Horas de insolação no mês de Janeiro

Distribuição	Valor de p	D
Cauchy	0.7057	0.098486
Qui Quadrado	0.1689	0.15565
Exponencial	3.601e-12	0.5149
Gama	0.9855	0.063842
Log-Normal	0.9801	0.065771
Normal	0.9964	0.057031
Weibull	0.9835	0.064596

Fonte: Autor (2019)

Tabela 16 – Umidade relativa no mês de Janeiro

Distribuição	Valor de p	D
Cauchy	0.2679	0.14416
Qui Quadrado	0.04	0.20215
Exponencial	2.401e-12	0.52516
Gama	0.3734	0.13129
Log-Normal	0.3448	0.13451
Normal	0.4963	0.11892
Weibull	0.955	0.072417

Fonte: Autor (2019)

B.2 Fevereiro

Tabela 17 – Precipitação acumulada no mês de Fevereiro

Distribuição	Valor de p	D
Cauchy	0.4214	0.12199
Qui Quadrado	4.557e-09	0.43743
Exponencial	0.0203	0.21009
Gama	0.5489	0.11054
Log-Normal	0.1141	0.16594
Normal	0.8342	0.086227
Weibull	0.9	0.079212

Fonte: Autor (2019)

Tabela 18 – Dias com precipitação no mês de Fevereiro

Distribuição	Valor de p	D
Cauchy	0.5828	0.15228
Qui Quadrado	0.08623	0.41796
Exponencial	0.001335	0.37499
Gama	0.6054	0.14963
Log-Normal	0.4584	0.1676
Normal	0.8138	0.12467
Weibull	0.8219	0.1236

Fonte: Autor (2019)

Tabela 19 – Temperatura mínima no mês de Fevereiro

Distribuição	Valor de p	D
Cauchy	0.3628	0.13042
Qui Quadrado	2.559e-06	0.36836
Exponencial	1.554e-15	0.59002
Gama	0.3116	0.13621
Log Normal	0.3369	0.13328
Normal	0.247	0.14456
Weibull	0.09361	0.17498

Fonte: Autor (2019)

Tabela 20 – Temperatura máxima no mês de Fevereiro

Distribuição	Valor de p	D
Cauchy	0.4788	0.11359
Qui Quadrado	9.652e-07	0.3657
Exponencial	2.2e-16	0.6026
Gama	0.8436	0.082364
Log Normal	0.9026	0.075985
Normal	0.7918	0.087198
Weibull	0.2153	0.14315

Fonte: Autor (2019)

Tabela 21 – Horas de insolação no mês de Fevereiro

Distribuição	Valor de p	D
Cauchy	0.6489	0.10424
Qui Quadrado	0.03285	0.20271
Exponencial	5.559e-10	0.46908
Gama	0.54	0.1135
Log-Normal	0.4198	0.12457
Normal	0.8059	0.090638
Weibull	0.8475	0.0866

Fonte: Autor (2019)

Tabela 22 – Umidade relativa no mês de Fevereiro

Distribuição	Valor de p	D
Cauchy	0.2228	0.14922
Qui Quadrado	0.04095	0.19943
Exponencial	9.57e-13	0.52802
Gama	0.601	0.10834
Log-Normal	0.5679	0.11123
Normal	0.7452	0.095817
Weibull	0.9004	0.080054

Fonte: Autor (2019)

B.3 Março

Tabela 23 – Precipitação acumulada no mês de Março

Distribuição	Valor de p	D
Cauchy	0.1926	0.14999
Qui Quadrado	1.767e-05	0.3345
Exponencial	0.01752	0.21343
Gama	0.893	0.080038
Log-Normal	0.9408	0.073598
Normal	0.2666	0.13912
Weibull	0.8827	0.081215

Fonte: Autor (2019)

Tabela 24 – Dias com precipitação no mês de Março

Distribuição	Valor de p	D
Cauchy	0.9089	0.11045
Qui Quadrado	0.6252	0.14731
Exponencial	0.003313	0.35091
Gama	0.8992	0.11215
Log-Normal	0.8586	0.1185
Normal	0.5855	0.15197
Weibull	0.6194	0.148

Fonte: Autor (2019)

Tabela 25 – Temperatura mínima no mês de Março

Distribuição	Valor de p	D
Cauchy	0.8624	0.085057
Qui Quadrado	6.933e-06	0.35457
Exponencial	6.661e-15	0.57738
Gama	0.9495	0.073569
Log Normal	0.9759	0.06771
Normal	0.9016	0.080583
Weibull	0.3695	0.12971

Fonte: Autor (2019)

Tabela 26 – Temperatura máxima no mês de Março

Distribuição	Valor de p	D
Cauchy	0.9053	0.075647
Qui Quadrado	4.36e-07	0.37531
Exponencial	2.2e-16	0.58833
Gama	0.6071	0.10265
Log Normal	0.5781	0.10504
Normal	0.6704	0.097454
Weibull	0.5584	0.1067

Fonte: Autor (2019)

Tabela 27 – Horas de insolação no mês de Março

Distribuição	Valor de p	D
Cauchy	0.5968	0.10448
Qui Quadrado	0.1106	0.16508
Exponencial	1.776e-12	0.50319
Gama	0.974	0.064685
Log-Normal	0.9658	0.066688
Normal	0.9745	0.064529
Weibull	0.9619	0.067532

Fonte: Autor (2019)

Tabela 28 – Umidade relativa no mês de Março

Distribuição	Valor de p	D
Cauchy	0.8533	0.084609
Qui Quadrado	0.005132	0.24469
Exponencial	1.665e-15	0.58029
Gama	0.7581	0.093704
Log-Normal	0.8259	0.08739
Normal	0.6981	0.098945
Weibull	0.1998	0.15139

Fonte: Autor (2019)

B.4 Julho

Tabela 29 – Precipitação acumulada no mês de julho

Distribuição	Valor de p	D
Cauchy	0.3067	0.13287
Qui Quadrado	1.812e-09	0.44321
Exponencial	0.02161	0.20667
Gama	0.7233	0.095152
Log Normal	0.2292	0.14291
Normal	0.1818	0.15039
Weibull	0.7626	0.091846

Fonte: Autor (2019)

Tabela 30 – Dias com precipitação em Julho

Distribuição	Valor de p	D
Cauchy	0.459	0.16142
Qui Quadrado	0.09778	0.389726
Exponencial	0.001054	0.36715
Gama	0.933	0.10196
Log-Normal	0.9022	0.10759
Normal	0.826	0.11858
Weibull	0.9198	0.10451

Fonte: Autor (2019)

Tabela 31 – Temperatura mínima no mês de Julho

Distribuição	Valor de p	D
Cauchy	0.5538	0.11461
Qui Quadrado	0.007228	0.24202
Exponencial	3.034e-08	0.43306
Gama	0.8442	0.088726
Log Normal	0.7187	0.10039
Normal	0.9508	0.074846
Weibull	0.9734	0.069828

Fonte: Autor (2019)

Tabela 32 – Temperatura máxima no mês de Julho

Distribuição	Valor de p	D
Cauchy	0.7255	0.099795
Qui Quadrado	0.000151	0.31444
Exponencial	2.218e-13	0.55743
Gama	0.9701	0.070687
Log Normal	0.959	0.073246
Normal	0.9424	0.074846
Weibull	0.5236	0.11731

Fonte: Autor (2019)

Tabela 33 – Horas de insolação no mês de Julho

Distribuição	Valor de p	D
Cauchy	0.6091	0.11217
Qui Quadrado	0.00408	0.25949
Exponencial	9.04e-08	0.42875
Gama	0.6606	0.10766
Log-Normal	0.4809	0.12383
Normal	0.9059	0.083446
Weibull	0.9698	0.072306

Fonte: Autor (2019)

Tabela 34 – Umidade relativa no mês de Julho

Distribuição	Valor de p	D
Cauchy	0.7069	0.10131
Qui Quadrado	0.001502	0.27724
Exponencial	7.883e-15	0.57855
Gama	0.9194	0.079162
Log-Normal	0.9138	0.07996
Normal	0.9329	0.07712
Weibull	0.4988	0.11997

Fonte: Autor (2019)

B.5 Agosto

Tabela 35 – Precipitação acumulada no mês de Agosto

Distribuição	Valor de p	D
Cauchy	0.1349	0.15947
Qui Quadrado	2.826e-08	0.41294
Exponencial	0.0293	0.19961
Gama	0.6309	0.10271
Log Normal	0.426	0.1204
Normal	0.4072	0.1222
Weibull	0.862	0.082652

Fonte: Autor (2019)

Tabela 36 – Dias com precipitação no mês de Agosto

Distribuição	Valor de p	D
Cauchy	0.459	0.16142
Qui Quadrado	0.09778	0.389726
Exponencial	0.001054	0.36715
Gama	0.933	0.10196
Log-Normal	0.9022	0.10759
Normal	0.826	0.11858
Weibull	0.9198	0.10451

Fonte: Autor (2019)

Tabela 37 – Temperatura mínima no mês de Agosto

Distribuição	Valor de p	D
Cauchy	0.485	0.11753
Qui Quadrado	0.0002403	0.3007
Exponencial	5.026e-12	0.50839
Gama	0.9873	0.061945
Log Normal	0.9417	0.073346
Normal	0.9817	0.064195
Poisson	0.01826	0.21705
Weibull	0.8646	0.083407

Fonte: Autor (2019)

Tabela 38 – Temperatura máxima no mês de Agosto

Distribuição	Valor de p	D
Cauchy	0.5886	0.10932
Qui Quadrado	2.763e-05	0.33451
Exponencial	7.361e-14	0.55618
Gama	0.895	0.081386
Log Normal	0.916	0.078717
Normal	0.832	0.08815
Poisson	0.0002792	0.29794
Weibull	0.6786	0.10175

Fonte: Autor (2019)

Tabela 39 – Horas de insolação no mês de Agosto

Distribuição	Valor de p	D
Cauchy	0.64	0.10829
Qui Quadrado	0.0057	0.24969
Exponencial	5.977e-08	0.42932
Gama	0.5614	0.11515
Log-Normal	0.3938	0.13117
Normal	0.9104	0.081955
Weibull	0.9799	0.068579

Fonte: Autor (2019)

Tabela 40 – Umidade relativa no mês de Agosto

Distribuição	Valor de p	D
Cauchy	0.7069	0.10131
Qui Quadrado	0.001502	0.27724
Exponencial	7.883e-15	0.57855
Gama	0.9194	0.079162
Log-Normal	0.9138	0.07996
Normal	0.9329	0.07712
Weibull	0.4988	0.11997

Fonte: Autor (2019)

B.6 Setembro

Tabela 41 – Precipitação acumulada no mês de Setembro

Distribuição	Valor de p	D
Cauchy	0.1711	0.15086
Qui Quadrado	6.157e-08	0.40019
Exponencial	0.0007799	0.26959
Gama	0.966	0.067602
Log Normal	0.8865	0.079279
Normal	0.3435	0.12754
Poisson	2.153e-10	0.461
Weibull	0.8191	0.086025

Fonte: Autor (2019)

Tabela 42 – Dias com precipitação no mês de Setembro

Distribuição	Valor de p	D
Cauchy	0.6763	0.14137
Qui Quadrado	0.05089	0.36119
Exponencial	0.00127	0.37626
Gama	0.6152	0.14848
Log-Normal	0.6919	0.13955
Normal	0.2867	0.19313
Weibull	0.3244	0.18681

Fonte: Autor (2019)

Tabela 43 – Temperatura mínima no mês de Setembro

Distribuição	Valor de p	D
Cauchy	0.5216	0.11195
Qui Quadrado	0.0007584	0.27553
Exponencial	5.462e-14	0.53665
Gama	0.9809	0.063232
Log Normal	0.968	0.066819
Normal	0.9542	0.069722
Poisson	0.001535	0.26297
Weibull	0.5603	0.10859

Fonte: Autor (2019)

Tabela 44 – Temperatura máxima no mês de Setembro

Distribuição	Valor de p	D
Cauchy	0.7175	0.095638
Qui Quadrado	3.272e-06	0.35453
Exponencial	1.665e-15	0.57215
Gama	0.6222	0.10343
Log Normal	0.6826	0.098502
Normal	0.5301	0.11108
Poisson	5.025e-06	0.34878
Weibull	0.1262	0.16144

Tabela 45 – Horas de insolação no mês de Setembro

Distribuição	Valor de p	D
Cauchy	0.6761	0.10516
Qui Quadrado	0.02655	0.21442
Exponencial	1.284e-08	0.44797
Gama	0.4525	0.12523
Log-Normal	0.3147	0.14011
Normal	0.7312	0.10035
Weibull	0.9015	0.08312

Fonte: Autor (2019)

Tabela 46 – Umidade relativa no mês de Setembro

Distribuição	Valor de p	D
Cauchy	0.7122	0.10193
Qui Quadrado	0.01556	0.23032
Exponencial	2.983e-13	0.55449
Gama	0.9525	0.074488
Log-Normal	0.9364	0.077378
Normal	0.9677	0.071109
Weibull	0.9687	0.070865

Fonte: Autor (2019)

APÊNDICE C – CÓDIGO FONTE

Clima.cs

```
/// <summary>
/// Método clima.
/// </summary>
/// <remarks>
/// Este método contém todas as funcionalidades do clima.
/// </remarks>
public class Clima{

    //Variáveis inicias do clima
    private float temperatura;
    private float temperaturamin;
    private float temperaturamax;
    private float variadortemp;
    private float precipitacao;
    private float precipitacaomensal;
    private float precipitacaodiaria;
    private float luminosidade;
    private float umidade;
    private float insolacaomensal;
    private int diasprecipitacao;
    private int enos;

    private float precipitacaoAcumulada;

    /// <summary>
    /// Construtor da classe.
    /// </summary>
    /// <param name="temperatura">Temperatura atual.</param>
    /// <param name="precipitacao">Precipitação média atual.</param>
    /// <param name="luminosidade">Radiação solar média atual.</param>
    /// <param name="diasprecipitacao">Quantidade de dias com precipitação no mês.</param>
    public Clima(float temperatura, float precipitacao, int luminosidade, float precipitacaomensal, int diasprecipitacao, float precipitacaodiaria)
    {
        this.temperatura = temperatura;
        this.precipitacao = precipitacao;
        this.diasprecipitacao = diasprecipitacao;
        this.luminosidade = luminosidade;
        this.precipitacaomensal = precipitacaomensal;
        this.precipitacaodiaria = precipitacaodiaria;
        this.precipitacaoAcumulada = 0;
    }

    /// <summary>
    /// Get ENOS.
    /// </summary>
    /// <returns>
    /// Retorna a ocorrencia ou não do El Niño Oscilação Sul.
    /// </returns>
    public int getEnos()
    {
        return this.enos;
    }


    /// <summary>
    /// Get Temperatura.
    /// </summary>
    /// <returns>
    /// Retorna a temperatura atual.
    /// </returns>
    public float getTemperatura(){
        return this.temperatura;
    }

    /// <summary>
    /// Get Temperatura mínima.
    /// </summary>
    /// <returns>
    /// Retorna a temperatura mínima no mês.
```

```
72          /// </returns>
73          ///
74          public float getTemperaturaMin()
75          {
76              return this.temperaturamin;
77          }
78
79
80          /// <summary>
81          /// Get Temperatura máxima.
82          /// </summary>
83          /// <returns>
84          /// Retorna a temperatura máxima no mês.
85          /// </returns>
86          ///
87
88          public float getTemperaturaMax()
89          {
90              return this.temperaturamax;
91          }
92
93
94          /// <summary>
95          /// Get variador de temperatura.
96          /// </summary>
97          /// <returns>
98          /// Retorna a variavel utilizada para variar a temperatura.
99          /// </returns>
100         ///
101         public float getVariadortemp()
102         {
103             return this.variadortemp;
104         }
105
106         /// <summary>
107         /// Get umidade.
108         /// </summary>
109         /// <returns>
110         /// Retorna a umidade.
111         /// </returns>
112         ///
113
114         public float getUmidade()
115         {
116             return this.umidade;
117         }
118
119         /// <summary>
120         /// Get Precipitação.
121         /// </summary>
122         /// <returns>
123         /// Retorna a preciputação horária.
124         /// </returns>
125         ///
126
127         public float getPrecipitacao(){
128             return this.precipitacao;
129         }
130
131
132         /// <summary>
133         /// Get Precipitação diaria.
134         /// </summary>
135         /// <returns>
136         /// Retorna a preciputação diaria.
137         /// </returns>
138         public float getPrecipitacaoDiaria()
139         {
140             return this.precipitacaodiaria;
141         }
142
143         /// <summary>
144         /// Get Dias Precipitação.
```

```
145         /// </summary>
146         /// <returns>
147         /// Retorna a quantidade de dias com preciputação.
148         /// </returns>
149         public int getDiasPrecipitacao()
150         {
151             return this.diasprecipitacao;
152         }
153 
154         /// <summary>
155         /// Get Precipitação Mensal.
156         /// </summary>
157         /// <returns>
158         /// Retorna a precipitação total para o mês.
159         /// </returns>
160         public float getPrecipitacaoMensal()
161         {
162             return this.precipitacaomensal;
163         }
164 
165 
166 
167         /// <summary>
168         /// Get Luminosidade.
169         /// </summary>
170         /// <returns>
171         /// Retorna a luminosidade diaria.
172         /// </returns>
173         public float getLuminosidade(){
174             return this.luminosidade;
175         }
176 
177         /// <summary>
178         /// Get Luminosidade.
179         /// </summary>
180         /// <returns>
181         /// Retorna luminosidade mensal.
182         /// </returns>
183         ///
184         public float getLuminosidadeMensal()
185         {
186             return this.insolacaomensal;
187         }
188 
189         /// <summary>
190         /// Get Precipitação Acumulada.
191         /// </summary>
192         /// <returns>
193         /// Retorna o somatório das precipitações diárias do mês.
194         /// </returns>
195         public float getPrecipitacaoAcumulada(){
196             return this.precipitacaoAcumulada;
197         }
198 
199 
200 
201         /// <summary>
202         /// Set ENOS.
203         /// </summary>
204         /// <param name="enos">Temperatura atual</param>
205         public void setEnos(int enos)
206         {
207             this.enos = enos;
208         }
209 
210         /// <summary>
211         /// Set Temperatura.
212         /// </summary>
213         /// <param name="temperatura">Temperatura atual</param>
214         public void setTemperatura(float temperatura){
215             this.temperatura = temperatura;
216         }
217 
```

```
218         /// <summary>
219         /// Set Temperatura mínima.
220         /// </summary>
221         /// <param name="temperaturamin">Temperatura mínima para o dia</param>
222         public void setTemperaturaMin(float temperaturamin)
223         {
224             this.temperaturamin = temperaturamin;
225         }
226 
227         /// <summary>
228         /// Set Temperatura máxima.
229         /// </summary>
230         /// <param name="temperaturamax">Temperatura máxima para o dia</param>
231         ///
232         public void setTemperaturaMax(float temperaturamax)
233         {
234             this.temperaturamax = temperaturamax;
235         }
236 
237         /// <summary>
238         /// Set variador de temperatura.
239         /// </summary>
240         /// <param name="variadortemp">Variador de temperatura</param>
241         public void setVariadorTemp(float variadortemp)
242         {
243             this.variadortemp = variadortemp;
244         }
245 
246         /// <summary>
247         /// Set umidade.
248         /// </summary>
249 
250             /// <param name="umidade">Umidade</param>
251         public void setUmidade(float umidade)
252         {
253             this.umidade = umidade;
254         }
255 
256 
257 
258 
259         /// <summary>
260         /// Set Precipitação.
261         /// </summary>
262         /// <param name="precipitacao">Precipitação atual.</param>
263         public void setPrecipitacao(float precipitacao){
264             this.precipitacao = precipitacao;
265         }
266 
267         /// <summary>
268         /// Set Precipitação diaria.
269         /// </summary>
270         /// <param name="precipitacaodiaria">Precipitação diaria.</param>
271         public void setPrecipitacaoDiaria(float precipitacaodiaria)
272         {
273             this.precipitacaodiaria = precipitacaodiaria;
274         }
275 
276 
277 
278         /// <summary>
279         /// Set Precipitação Mensal.
280         /// </summary>
281         /// <param name="precipitacaomensal">Precipitação mensal.</param>
282         public void setPrecipitacaoMensal(float precipitacaomensal)
283         {
284             this.precipitacaomensal = precipitacaomensal;
285         }
286 
287 
288         /// <summary>
289         /// Set Dias com Precipitação.
290         /// </summary>
```

```
291         /// <param name="diasprecipitacao"> Dias com precipitação.</param>
292         public void setDiasPrecipitacao(int diasprecipitacao)
293         {
294             this.diasprecipitacao = diasprecipitacao;
295         }
296 
297 
298 
299         /// <summary>
300         /// Set Luminosidade.
301         /// </summary>
302         /// <param name="luminosidade">Luminosidade.</param>
303         public void setLuminosidade(float luminosidade){
304             this.luminosidade = luminosidade;
305         }
306 
307         /// <summary>
308         /// Set Luminosidade Mensal.
309         /// </summary>
310         /// <param name="luminosidade">Luminosidade Mensal.</param>
311 
312         public void setLuminosidadeMensal(float insolacaomensal)
313         {
314             this.insolacaomensal = insolacaomensal;
315         }
316 
317 
318 
319 
320         /// <summary>
321         /// Zerar a precipitação acumulada.
322         /// </summary>
323         /// <remarks>
324         /// Zera a precipitação acumulada. Chamar este método todo final de mês.
325         /// </remarks>
326         public void zerarPrecipitacaoAcumulada(){
327             precipitacaoAcumulada = 0;
328         }
329 
330         /// <summary>
331         /// Ìncrementar a precipitação acumulada.
332         /// </summary>
333         /// <remarks>
334         /// Incrementa o valor da precipitação acumulada.
335         /// </remarks>
336         /// <param name="valor">Precipitação atual.</param>
337         public void adicionarPrecipitacaoAcumulada(float valor){
338             precipitacaoAcumulada += valor;
339         }
340 
341     }
```

ModuloClimatico.cs

```
using UnityEngine;
using GeradorValores;
using System;
/// <summary>
/// Módulo externo climático.
/// </summary>
/// <remarks>
/// Módulo climático para o jogo SIMCOW.
/// </remarks>
public static class ModuloClimatico{


    /// <summary>
    /// Método que retorna a temperatura mínima para o dia.
    /// </summary>
    /// <param name="mes">Mês atual.</param>
    /// <returns></returns>
    public static float calcularTemperaturaMin(int mes, int enos){
        double tempminimad = 0;
        float temperaturamin = 0;

        if (mes == 1) //Janeiro
        {
            tempminimad = GeradorAleatorio.DistLogNormal(2.907902093, 0.054696393);

        }
        if (mes == 2) //Fevereiro
        {
            tempminimad = GeradorAleatorio.DistCauchy(17.8786492, 0.5866537);

        }
        if (mes == 3)//Março
        {
            tempminimad = GeradorAleatorio.DistLogNormal(2.811323749, 0.068106696);
        }
        if (mes == 4)//Abril
        {
            if (enos == 0) // Sem ocorrência
            {
                tempminimad = GeradorAleatorio.DistLogNormal(2.60463877, 0.07734995);

            }
            if (enos == 1)// El Niño
            {
                tempminimad = GeradorAleatorio.DistCauchy(14.5589570, 0.8011363);

            }

            if (enos == 2)// La Niña
            {
                tempminimad = GeradorAleatorio.DistCauchy(12.6599951, 0.8468830);

            }


        }
        if (mes == 5)//Maio
        {

            if (enos == 0) // Sem ocorrência
            {
                tempminimad = GeradorAleatorio.DistGama(160.19035, 7.60524);

            }
            if (enos == 1)// El Niño
            {
                tempminimad = GeradorAleatorio.DistCauchy(11.2219652, 0.4282306);

            }

            if (enos == 2)// La Niña
```

```
74                  {
75                      tempminimad = GeradorAleatorio.DistCauchy(10.2987988, 0.8043056);
76
77                  }
78
79
80
81
82              }
83              if (mes == 6)//Junho
84              {
85                  if (enos == 0) // Sem ocorrência
86                  {
87                      tempminimad = GeradorAleatorio.DistCauchy(9.0772070, 0.8414665);
88
89                  }
90                  if (enos == 1)// El Niño
91                  {
92                      tempminimad = GeradorAleatorio.DistWeibull(6.4981662, 9.2628866);
93
94                  }
95
96                  if (enos == 2)// La Niña
97                  {
98                      tempminimad = GeradorAleatorio.DistCauchy(7.3216653, 0.6110179);
99
100                 }
101             }
102             if (mes == 7)//Julho
103             {
104
105                 tempminimad = GeradorAleatorio.DistWeibull(1.4068971, 145.5241196);
106
107             }
108             if (mes == 8)//Agosto
109             {
110                 tempminimad = GeradorAleatorio.DistGama(2.338168526, 0.020854023);
111             }
112             if (mes == 9)//Setembro
113             {
114                 tempminimad = GeradorAleatorio.DistGama(2.888354860, 0.022194396);
115             }
116             if (mes == 10)//Outubro
117             {
118                 if (enos == 0) // Sem ocorrência
119                 {
120                     tempminimad = GeradorAleatorio.DistWeibull(8.9133060, 13.4973222);
121
122                 }
123                 if (enos == 1)// El Niño
124                 {
125
126                     tempminimad = GeradorAleatorio.DistWeibull(13.3913257, 13.1873280);
127                 }
128
129                 if (enos == 2)// La Niña
130                 {
131                     tempminimad = GeradorAleatorio.DistNormal(12.6487456, 1.2433617);
132
133                 }
134             }
135             if (mes == 11)//Novembro
136             {
137                 if (enos == 0) // Sem ocorrência
138                 {
139                     tempminimad = GeradorAleatorio.DistNormal(14.9271136, 1.1521471);
140
141                 }
142                 if (enos == 1)// El Niño
143                 {
144                     tempminimad = GeradorAleatorio.DistNormal(12.6774055, 1.1494616);
145
146                 }
```

```
147
148                 if (enos == 2)// La Niña
149                 {
150                     tempminimad = GeradorAleatorio.DistWeibull(9.9498952, 13.2419561);
151
152                 }
153
154
155             }
156             if (mes == 12)//Dezembro
157             {
158                 if (enos == 0) // Sem ocorrência
159                 {
160                     tempminimad = GeradorAleatorio.DistLogNormal(2.829512040,
                        0.063865814);
161
162                 }
163                 if (enos == 1)// El Niño
164                 {
165                     tempminimad = GeradorAleatorio.DistGama(453.561948, 26.283096);
166
167                 }
168
169                 if (enos == 2)// La Niña
170                 {
171                     tempminimad = GeradorAleatorio.DistLogNormal(2.81395878, 0.05272971);
172
173                 }
174             }
175
176
177              temperaturamin = Convert.ToSingle(tempminimad);
178
179             return temperaturamin;
180         }
181
182         /// <summary>
183         /// Método que calcula a temperatura máxima.
184         /// </summary>
185         /// <param name="mes">Mês.</param>
186         /// <returns></returns>
187         public static float calcularTemperaturaMax(int mes, int enos)
188         {
189             double tempmaximad = 0;
190
191             if (mes == 1) //Janeiro
192             {
193                 tempmaximad = GeradorAleatorio.DistLogNormal(3.402052171, 0.038945757);
194
195             }
196             if (mes == 2) //Fevereiro
197             {
198                 tempmaximad = GeradorAleatorio.DistLogNormal(3.373868981, 0.047043264);
199
200             }
201             if (mes == 3)//Março
202             {
203                 tempmaximad = GeradorAleatorio.DistCauchy(27.5703457, 0.6929439);
204             }
205             if (mes == 4)//Abril
206             {
207                 if (enos == 0) // Sem ocorrência
208                 {
209                     tempmaximad = GeradorAleatorio.DistLogNormal(24.0935911, 0.6737199);
210
211                 }
212                 if (enos == 1)// El Niño
213                 {
214                     tempmaximad = GeradorAleatorio.DistCauchy(23.7992257, 0.8767212);
215
216                 }
217
218                 if (enos == 2)// La Niña
```

```
219                 {
220                     tempmaximad = GeradorAleatorio.DistCauchy(24.2579231, 2.0358531);
221 
222                 }
223 
224 
225             }
226             if (mes == 5)//Maio
227             {
228 
229                 if (enos == 0) // Sem ocorrência
230                 {
231                     tempmaximad = GeradorAleatorio.DistNormal(21.0631644, 1.6318821);
232 
233                 }
234                 if (enos == 1)// El Niño
235                 {
236                     tempmaximad = GeradorAleatorio.DistCauchy(19.8147222, 0.7840291);
237 
238                 }
239 
240                 if (enos == 2)// La Niña
241                 {
242                     tempmaximad = GeradorAleatorio.DistGama(221.748609, 11.042797);
243 
244                 }
245 
246 
247 
248 
249             }
250             if (mes == 6)//Junho
251             {
252                 if (enos == 0) // Sem ocorrência
253                 {
254                     tempmaximad = GeradorAleatorio.DistNormal(18.1273913, 1.4202483);
255 
256                 }
257                 if (enos == 1)// El Niño
258                 {
259                     tempmaximad = GeradorAleatorio.DistGama(179.752527, 10.198930);
260 
261                 }
262 
263                 if (enos == 2)// La Niña
264                 {
265                     tempmaximad = GeradorAleatorio.DistCauchy(17.0087186, 0.7784668);
266 
267                 }
268             }
269             if (mes == 7)//Julho
270             {
271 
272                 tempmaximad = GeradorAleatorio.DistGama(102.834750, 5.871275);
273 
274             }
275             if (mes == 8)//Agosto
276             {
277                 tempmaximad = GeradorAleatorio.DistLogNormal(2.953441986, 0.084307765);
278             }
279             if (mes == 9)//Setembro
280             {
281                 tempmaximad = GeradorAleatorio.DistCauchy(20.5585482, 0.7409044);
282             }
283             if (mes == 10)//Outubro
284             {
285                 if (enos == 0) // Sem ocorrência
286                 {
287                     tempmaximad = GeradorAleatorio.DistGama(424.167191, 18.002780);
288 
289                 }
290                 if (enos == 1)// El Niño
291                 {
```

```
292
293                     tempmaximad = GeradorAleatorio.DistCauchy(23.0952515, 0.5429098);
294                 }
295
296                 if (enos == 2)// La Niña
297                 {
298                     tempmaximad = GeradorAleatorio.DistNormal(23.5820467, 0.9225360);
299
300                 }
301             }
302             if (mes == 11)//Novembro
303             {
304                 if (enos == 0) // Sem ocorrência
305                 {
306                     tempmaximad = GeradorAleatorio.DistWeibull(26.8181552, 27.1288207);
307
308                 }
309                 if (enos == 1)// El Niño
310                 {
311                     tempmaximad = GeradorAleatorio.DistCauchy(25.3861988, 0.6268056);
312
313                 }
314
315                 if (enos == 2)// La Niña
316                 {
317                     tempmaximad = GeradorAleatorio.DistLogNormal(26.2430371, 1.3320219);
318
319                 }
320
321
322             }
323             if (mes == 12)//Dezembro
324             {
325                 if (enos == 0) // Sem ocorrência
326                 {
327                     tempmaximad = GeradorAleatorio.DistLogNormal(3.370186228,
                        0.055720334);
328
329                 }
330                 if (enos == 1)// El Niño
331                 {
332                     tempmaximad = GeradorAleatorio.DistLogNormal(3.351262557,
                        0.037386777);
333
334                 }
335
336                 if (enos == 2)// La Niña
337                 {
338                     tempmaximad = GeradorAleatorio.DistCauchy(29.3835693, 1.1281388);
339
340                 }
341             }
342
343
344
345
346
347             float temperaturamax = Convert.ToSingle(tempmaximad);
348
349             return temperaturamax;
350         }
351
352         /// <summary>
353         /// Método que calcula a temperatura momentanea.
354         /// </summary>
355         /// <param name="temperatura">Temperatura.</param>
356         /// <param name="variador">Variador de temperatura.</param>
357         /// <param name="hora">Hora.</param>
358         /// <returns></returns>
359         public static float calcularTemperatura(float temperatura, float variador, int
            hora)
360         {
361
```

```
362             if (hora <= 6)
363             {
364                 temperatura = temperatura - variador;
365 
366             }
367 
368 
369             else if (hora > 6 &&  hora <= 18)
370             {
371                  temperatura = temperatura +  (2* variador);
372             }
373 
374             else if(hora > 18)
375             {
376 
377                 temperatura = temperatura - ( variador);
378             }
379 
380             return temperatura;
381 
382 
383         }
384 
385         /// <summary>
386         /// Método que calcula o variador de temperatura.
387         /// </summary>
388         /// <param name="tempmin">Temperatura mínima.</param>
389         /// <param name="tempmax">Temperatura máxima.</param>
390         /// <returns></returns>
391         public static float calcularVariadorTemp(float tempmin, float tempmax)
392         {
393 
394             float variadortemp = (tempmax-tempmin) / 24;
395             return variadortemp;
396 
397 
398         }
399 
400 
401         /// <summary>
402         /// Método que calcula a precipitação mensal.
403         /// </summary>
404         /// <param name="mes">Mês atual.</param>
405         /// <returns></returns>
406 
407 
408         public static float calcularPrecipitacaoMensal(int mes, int enos)
409         {
410             double precipitacaomensal = 0;
411 
412 
413 
414 
415             if (mes == 1) //Janeiro
416             {
417                 precipitacaomensal = GeradorAleatorio.DistGama(2.447615026, 0.020687878);
418 
419             }
420             if (mes == 2) //Fevereiro
421             {
422                 precipitacaomensal = GeradorAleatorio.DistWeibull(1.6342041, 140.0681606);
423 
424             }
425             if (mes == 3)//Março
426             {
427                 precipitacaomensal = GeradorAleatorio.DistLogNormal(4.50375895,
                    0.70645960);
428             }
429             if (mes == 4)//Abril
430             {
431                 if (enos == 0) // Sem ocorrência
432                 {
433                     precipitacaomensal = GeradorAleatorio.DistWeibull(1.4742512,
```

```
                    132.0851798);
434
435             }
436             if (enos == 1)// El Niño
437             {
438                 precipitacaomensal = GeradorAleatorio.DistWeibull(1.4742512,
                    153.1478032);
439
440             }
441
442             if (enos == 2)// La Niña
443             {
444                 precipitacaomensal = GeradorAleatorio.DistCauchy(103.924881,
                    20.508820);
445
446             }
447
448
449         }
450         if (mes == 5)//Maio
451         {
452
453             if (enos == 0) // Sem ocorrência
454             {
455                 precipitacaomensal = GeradorAleatorio.DistNormal(114.21852, 74.12098);
456
457             }
458             if (enos == 1)// El Niño
459             {
460                 precipitacaomensal = GeradorAleatorio.DistNormal(113.15000, 86.00746);
461
462             }
463
464             if (enos == 2)// La Niña
465             {
466                 precipitacaomensal = GeradorAleatorio.DistCauchy(76.732485,
                    17.700734);
467
468             }
469
470
471
472
473         }
474         if (mes == 6)//Junho
475         {
476             if (enos == 0) // Sem ocorrência
477             {
478                 precipitacaomensal = GeradorAleatorio.DistWeibull(1.8291662,
                    111.3732370);
479
480             }
481             if (enos == 1)// El Niño
482             {
483                 precipitacaomensal = GeradorAleatorio.DistCauchy(10.8476750,
                    1.9310272);
484
485             }
486
487             if (enos == 2)// La Niña
488             {
489                 precipitacaomensal = GeradorAleatorio.DistCauchy(102.32266, 31.76442);
490
491             }
492         }
493         if (mes == 7)//Julho
494         {
495
496             precipitacaomensal = GeradorAleatorio.DistWeibull(1.4068971, 145.5241196);
497
498         }
499         if (mes == 8)//Agosto
500         {
```

```
501                 precipitacaomensal = GeradorAleatorio.DistWeibull(1.6962704, 125.9296593);
502             }
503             if (mes == 9)//Setembro
504             {
505                 precipitacaomensal = GeradorAleatorio.DistGama(2.888354860, 0.022194396);
506             }
507             if (mes == 10)//Outubro
508             {
509                 if (enos == 0) // Sem ocorrência
510                 {
511                     precipitacaomensal = GeradorAleatorio.DistNormal(143.844828,
                        66.618132);
512 
513                 }
514                 if (enos == 1)// El Niño
515                 {
516 
517                     precipitacaomensal = GeradorAleatorio.DistLogNormal(5.03921745,
                        0.52647121);
518                 }
519 
520                 if (enos == 2)// La Niña
521                 {
522                     precipitacaomensal = GeradorAleatorio.DistCauchy(124.49778, 30.13661);
523 
524                 }
525             }
526             if (mes == 11)//Novembro
527             {
528                 if (enos == 0) // Sem ocorrência
529                 {
530                     precipitacaomensal = GeradorAleatorio.DistWeibull(1.954736,
                        110.894595);
531 
532                 }
533                 if (enos == 1)// El Niño
534                 {
535                     precipitacaomensal = GeradorAleatorio.DistCauchy(131.779879,
                        22.729950);
536 
537                 }
538 
539                 if (enos == 2)// La Niña
540                 {
541                     precipitacaomensal = GeradorAleatorio.DistGama(2.69168366,
                        0.03943270);
542 
543                 }
544 
545 
546             }
547             if (mes == 12)//Dezembro
548             {
549                 if (enos == 0) // Sem ocorrência
550                 {
551                     precipitacaomensal = GeradorAleatorio.DistNormal(99.60, 68.87);
552 
553                 }
554                 if (enos == 1)// El Niño
555                 {
556                     precipitacaomensal = GeradorAleatorio.DistLogNormal(4.8490342,
                        0.6746017);
557 
558                 }
559 
560                 if (enos == 2)// La Niña
561                 {
562                     precipitacaomensal = GeradorAleatorio.DistCauchy(58.221147,
                        13.300121);
563 
564                 }
565             }
566 
```

```
567             float precipitacaomes = Convert.ToSingle(precipitacaomensal);
568 
569             return precipitacaomes;
570         }
571 
572 
573 
574         /// <summary>
575         /// Método que calcula a precipitação atual.
576         /// </summary>
577         /// <param name="precipitacaodiaria">Precipitação diária.</param>
578         /// <param name="precipitacaomensal">Precipitação total para o mês.</param>
579         /// <param name="acumulada">Chuva acumulada no mês até determinado
            momento.</param>
580         /// <returns></returns>
581 
582         public static float calcularPrecipitacao( float precipitacaodiaria, float
            precipitacaomensal, float acumulada)
583         {
584             float precipitacaohora = precipitacaodiaria/10;
585             float precipitacao = 0;
586 
587             if (System.DateTime.Now.Millisecond % 2 == 0)
588             {
589 
590                 if (acumulada <= precipitacaomensal)
591 
592                 {
593                     precipitacao = UnityEngine.Random.Range(0, precipitacaohora);
594                 }
595 
596                 else {
597                     precipitacao = 0;
598                 }
599             }
600 
601             else {
602                 precipitacao = 0;
603             }
604 
605             return precipitacao;
606         }
607 
608 
609 
610         /// <summary>
611         /// Método que retorna a precipitação diária.
612         /// </summary>
613         /// <param name="dias">Dias com precipitação.</param>
614         /// <param name="precipitacaomensal">Precipitação total calculada para o
            mês.</param>
615         /// <param name="acumulada">Precipitação acumulada até determinado momento</param>
616         /// <returns></returns>
617 
618         public static float calcularPrecipitacaoDiaria(int dias, float
            precipitacaomensal, float acumulada)
619         {
620             float precipitacaodiaria;
621 
622             if (System.DateTime.Now.Millisecond % 2 == 0)
623             {
624 
625 
626             if (acumulada <= precipitacaomensal)
627 
628                 {
629 
630 
631                         precipitacaodiaria = precipitacaomensal / dias;
632 
633 
634                 }
635 
```

```
636                 else { precipitacaodiaria = 0; }
637             }
638             else { precipitacaodiaria = 0; }
639             return precipitacaodiaria;
640         }
641 
642 
643         /// <summary>
644         /// Método que retorna a quantidade de dias com precipitação.
645         /// </summary>
646         /// <param name="mes">Mês atual.</param>
647         /// <returns></returns>
648 
649         public static int calcularDiasPrecipitacao(int mes)
650         {
651                double diasprecipitacao = 0;
652 
653             int enos = 0;
654             if (mes == 1) //Janeiro
655             {
656                 diasprecipitacao = GeradorAleatorio.DistLogNormal(2.2782484, 0.3512322);
657 
658             }
659             if (mes == 2) //Fevereiro
660             {
661                 diasprecipitacao = GeradorAleatorio.DistWeibull(3.205686, 11.610705);
662 
663             }
664             if (mes == 3)//Março
665             {
666                 diasprecipitacao = GeradorAleatorio.DistLogNormal(2.2224526, 0.3967231);
667             }
668             if (mes == 4)//Abril
669             {
670                 if (enos == 0) // Sem ocorrência
671                 {
672                     diasprecipitacao = GeradorAleatorio.DistNormal(6.600000, 2.244994);
673 
674                 }
675                 if (enos == 1)// El Niño
676                 {
677                     diasprecipitacao = GeradorAleatorio.DistCauchy(9.768236, 1.631143);
678 
679                 }
680 
681                 if (enos == 2)// La Niña
682                 {
683                     diasprecipitacao = GeradorAleatorio.DistCauchy(6.7985204, 0.8151006);
684 
685                 }
686 
687 
688             }
689             if (mes == 5)//Maio
690             {
691 
692                 if (enos == 0) // Sem ocorrência
693                 {
694                     diasprecipitacao = GeradorAleatorio.DistNormal(8.466667, 3.685407);
695 
696                 }
697                 if (enos == 1)// El Niño
698                 {
699                     diasprecipitacao = GeradorAleatorio.DistCauchy(9.582647, 2.059009);
700 
701                 }
702 
703                 if (enos == 2)// La Niña
704                 {
705                     diasprecipitacao = GeradorAleatorio.DistLogNormal(2.1140421,
                        0.2209493);
706 
707                 }
```

```
708
709
710
711
712             }
713             if (mes == 6)//Junho
714             {
715                 if (enos == 0) // Sem ocorrência
716                 {
717                     diasprecipitacao = GeradorAleatorio.DistLogNormal(2.2233628,
                        0.3832615);
718
719                 }
720                 if (enos == 1)// El Niño
721                 {
722                     diasprecipitacao = GeradorAleatorio.DistNormal(11.0, 3.162278);
723
724                 }
725
726                 if (enos == 2)// La Niña
727                 {
728                     diasprecipitacao = GeradorAleatorio.DistLogNormal(2.365664, 0.135189);
729
730                 }
731             }
732             if (mes == 7)//Julho
733             {
734
735                 diasprecipitacao = GeradorAleatorio.DistWeibull(2.734148, 12.328236);
736
737             }
738             if (mes == 8)//Agosto
739             {
740                 diasprecipitacao = GeradorAleatorio.DistWeibull(2.734148, 12.328236);
741             }
742             if (mes == 9)//Setembro
743             {
744                 diasprecipitacao = GeradorAleatorio.DistGama(8.7299008, 0.8437825);
745             }
746             if (mes == 10)//Outubro
747             {
748                 if (enos == 0) // Sem ocorrência
749                 {
750                     diasprecipitacao = GeradorAleatorio.DistGama(11.579204, 1.240629);
751
752                 }
753                 if (enos == 1)// El Niño
754                 {
755
756                     diasprecipitacao = GeradorAleatorio.DistGama(5.1659486, 0.5397253);
757                 }
758
759                 if (enos == 2)// La Niña
760                 {
761                     diasprecipitacao = GeradorAleatorio.DistCauchy(11.5, 1.936492);
762
763                 }
764             }
765             if (mes == 11)//Novembro
766             {
767                 if (enos == 0) // Sem ocorrência
768                 {
769                     diasprecipitacao = GeradorAleatorio.DistGama(7.1826671, 0.9310866);
770
771                 }
772                 if (enos == 1)// El Niño
773                 {
774                     diasprecipitacao = GeradorAleatorio.DistCauchy(10.127605, 1.638094);
775
776                 }
777
778                 if (enos == 2)// La Niña
779                 {
```

```
780                 diasprecipitacao = GeradorAleatorio.DistCauchy(7.413850, 1.324152);
781
782             }
783
784
785         }
786         if (mes == 12)//Dezembro
787         {
788             if (enos == 0) // Sem ocorrência
789             {
790                 diasprecipitacao = GeradorAleatorio.DistCauchy(7.8577710, 1.7753140);
791
792             }
793             if (enos == 1)// El Niño
794             {
795                 diasprecipitacao = GeradorAleatorio.DistCauchy(8.7017497, 1.0069021);
796
797             }
798
799             if (enos == 2)// La Niña
800             {
801                 diasprecipitacao = GeradorAleatorio.DistCauchy(7, 2);
802
803             }
804         }
805
806         int diasprecipitacaoint = Convert.ToInt32(diasprecipitacao);
807
808         return diasprecipitacaoint;
809     }
810
811
812
813         /// <summary>
814         /// Método que retorna a luminosidade atual.
815         /// </summary>
816         /// <param name="mes">Mês atual.</param>
817         /// <returns></returns>
818         public static float calcularLuminosidadeMensal(int mes, int enos){
819
820         double luminosidademensal = 0;
821
822
823         if (mes == 1) //Janeiro
824         {
825             luminosidademensal = GeradorAleatorio.DistNormal(243.111765, 31.664861);
826
827
828         }
829         if (mes == 2) //Fevereiro
830         {
831             luminosidademensal = GeradorAleatorio.DistWeibull(6.6811783, 222.7192788);
832
833         }
834         if (mes == 3)//Março
835         {
836             luminosidademensal = GeradorAleatorio.DistNormal(210.849, 32.13912);
837         }
838         if (mes == 4)//Abril
839         {
840             if (enos == 0) // Sem ocorrência
841             {
842                 luminosidademensal = GeradorAleatorio.DistCauchy(201.266891,
                    18.551802);
843
844             }
845             if (enos == 1)// El Niño
846             {
847                 luminosidademensal = GeradorAleatorio.DistWeibull(4.8111266,
                    175.9842569);
848
849             }
850
```

```
851                 if (enos == 2)// La Niña
852                 {
853                     luminosidademensal = GeradorAleatorio.DistWeibull(7.881309,
                        205.086236);
854 
855                 }
856 
857 
858             }
859             if (mes == 5)//Maio
860             {
861 
862                 if (enos == 0) // Sem ocorrência
863                 {
864                     luminosidademensal = GeradorAleatorio.DistWeibull(6.716698,
                        176.545045);
865 
866                 }
867                 if (enos == 1)// El Niño
868                 {
869                     luminosidademensal = GeradorAleatorio.DistCauchy(153.718209,
                        14.640684);
870 
871                 }
872 
873                 if (enos == 2)// La Niña
874                 {
875                     luminosidademensal = GeradorAleatorio.DistCauchy(157.634851,
                        17.312279);
876 
877                 }
878 
879 
880 
881 
882             }
883             if (mes == 6)//Junho
884             {
885                 if (enos == 0) // Sem ocorrência
886                 {
887                     luminosidademensal = GeradorAleatorio.DistCauchy(138.586933,
                        18.483002);
888 
889                 }
890                 if (enos == 1)// El Niño
891                 {
892                     luminosidademensal = GeradorAleatorio.DistNormal(124.365515,
                        20.780383);
893 
894                 }
895 
896                 if (enos == 2)// La Niña
897                 {
898                     luminosidademensal = GeradorAleatorio.DistCauchy(132.806181,
                        7.554498);
899 
900                 }
901             }
902             if (mes == 7)//Julho
903             {
904 
905                 luminosidademensal = GeradorAleatorio.DistWeibull(5.1816373, 154.4074254);
906 
907             }
908             if (mes == 8)//Agosto
909             {
910                 luminosidademensal = GeradorAleatorio.DistWeibull(5.2256040, 154.2465060);
911             }
912             if (mes == 9)//Setembro
913             {
914                 luminosidademensal = GeradorAleatorio.DistWeibull(6.3807679, 168.5655949);
915             }
916             if (mes == 10)//Outubro
```

```
917             {
918                 if (enos == 0) // Sem ocorrência
919                 {
920                     luminosidademensal = GeradorAleatorio.DistGama(6.5101771,
                        216.0348284);
921 
922                 }
923                 if (enos == 1)// El Niño
924                 {
925 
926                     luminosidademensal = GeradorAleatorio.DistWeibull(6.924635,
                        217.363454);
927                 }
928 
929                 if (enos == 2)// La Niña
930                 {
931                     luminosidademensal = GeradorAleatorio.DistWeibull(6.663111,
                        214.077162);
932 
933                 }
934             }
935             if (mes == 11)//Novembro
936             {
937                 if (enos == 0) // Sem ocorrência
938                 {
939                     luminosidademensal = GeradorAleatorio.DistNormal(9.573287,
                        246.292599);
940 
941                 }
942                 if (enos == 1)// El Niño
943                 {
944                     luminosidademensal = GeradorAleatorio.DistNormal(201.264286,
                        31.892176);
945 
946                 }
947 
948                 if (enos == 2)// La Niña
949                 {
950                     luminosidademensal = GeradorAleatorio.DistWeibull(8.976441,
                        271.207839);
951 
952                 }
953 
954 
955             }
956             if (mes == 12)//Dezembro
957             {
958                 if (enos == 0) // Sem ocorrência
959                 {
960                     luminosidademensal = GeradorAleatorio.DistLogNormal(5.55156583,
                        0.10690100);
961 
962                 }
963                 if (enos == 1)// El Niño
964                 {
965                     luminosidademensal = GeradorAleatorio.DistGama(64.1622725, 0.2834560);
966 
967                 }
968 
969                 if (enos == 2)// La Niña
970                 {
971                     luminosidademensal = GeradorAleatorio.DistLogNormal(5.55773584,
                        0.10365508);
972 
973                 }
974             }
975 
976             float luminosidademes = Convert.ToSingle(luminosidademensal);
977 
978             return luminosidademes;
979         }
980 
981         /// <summary>
```

```
982         /// Método que retorna a luminosidade Mensal.
983         /// </summary>
984         /// <param name="mes">Mês atual.</param>
985         /// <returns></returns>
986
987         public static float calcularLuminosidade(float insolacao, int mes)
988         {
989             int dias = 0;
990
991             if (mes == 1 || mes == 3 || mes == 5 || mes == 7 || mes == 8 || mes == 10 ||
                mes == 12)
992             {
993
994                  dias = 31;
995             }
996
997             else if(mes == 4 || mes == 6 || mes == 9 || mes == 11)
998             {
999
1000                 dias = 30;
1001
1002            }
1003            else if (mes == 2 )
1004            {
1005
1006                dias = 28;
1007
1008            }
1009
1010            float insolacaodiaria = insolacao / dias;
1011            return insolacaodiaria;
1012
1013        }
1014
1015
1016        /// <summary>
1017        /// Método que retorna a umidade atual.
1018        /// </summary>
1019        /// <param name="mes">Mês atual.</param>
1020        /// <returns></returns>
1021        public static float calcularUmidade(int mes, int enos)
1022        {
1023
1024            double umidade = 0;
1025
1026
1027            if (mes == 1) //Janeiro
1028            {
1029                umidade = GeradorAleatorio.DistWeibull(12.4851690, 68.0521433);
1030
1031
1032            }
1033            if (mes == 2) //Fevereiro
1034            {
1035                umidade = GeradorAleatorio.DistWeibull(12.106877, 71.837299);
1036
1037            }
1038            if (mes == 3)//Março
1039            {
1040                umidade = GeradorAleatorio.DistCauchy(70.1829053, 2.4627116);
1041            }
1042            if (mes == 4)//Abril
1043            {
1044                if (enos == 0) // Sem ocorrência
1045                {
1046                    umidade = GeradorAleatorio.DistWeibull(19.353937, 71.819854);
1047
1048                }
1049                if (enos == 1)// El Niño
1050                {
1051                    umidade = GeradorAleatorio.DistWeibull(13.622199, 77.040658);
1052
1053                }
```

```
1054
1055                if (enos == 2)// La Niña
1056                {
1057                    umidade = GeradorAleatorio.DistGama(412.001086, 5.873457);
1058
1059                }
1060
1061
1062            }
1063            if (mes == 5)//Maio
1064            {
1065
1066                if (enos == 0) // Sem ocorrência
1067                {
1068                    umidade = GeradorAleatorio.DistNormal(75.8641305, 5.2475202);
1069
1070                }
1071                if (enos == 1)// El Niño
1072                {
1073                    umidade = GeradorAleatorio.DistWeibull(21.734745, 80.148107);
1074
1075                }
1076
1077                if (enos == 2)// La Niña
1078                {
1079                    umidade = GeradorAleatorio.DistWeibull(7.087037, 171.618714);
1080
1081                }
1082
1083
1084
1085
1086            }
1087            if (mes == 6)//Junho
1088            {
1089                if (enos == 0) // Sem ocorrência
1090                {
1091                    umidade = GeradorAleatorio.DistWeibull(19.0894707, 80.0054439);
1092
1093                }
1094                if (enos == 1)// El Niño
1095                {
1096                    umidade = GeradorAleatorio.DistGama(3.8744787, 128.7308353);
1097
1098                }
1099
1100                if (enos == 2)// La Niña
1101                {
1102                    umidade = GeradorAleatorio.DistWeibull(20.827330, 77.378738);
1103
1104                }
1105            }
1106            if (mes == 7)//Julho
1107            {
1108
1109                umidade = GeradorAleatorio.DistNormal(67.516119, 3.2342511);
1110
1111            }
1112            if (mes == 8)//Agosto
1113            {
1114                umidade = GeradorAleatorio.DistNormal(76.8137992, 4.3253782);
1115            }
1116            if (mes == 9)//Setembro
1117            {
1118                umidade = GeradorAleatorio.DistWeibull(15.1586717, 75.5635863);
1119            }
1120            if (mes == 10)//Outubro
1121            {
1122                if (enos == 0) // Sem ocorrência
1123                {
1124                    umidade = GeradorAleatorio.DistWeibull(16.428376, 73.043990);
1125
1126                }
```

```
1127                if (enos == 1)// El Niño
1128                {
1129
1130                    umidade = GeradorAleatorio.DistGama(162.639850, 2.285032);
1131                }
1132
1133                if (enos == 2)// La Niña
1134                {
1135                    umidade = GeradorAleatorio.DistNormal(69.308377, 5.490516);
1136
1137                }
1138            }
1139            if (mes == 11)//Novembro
1140            {
1141                if (enos == 0) // Sem ocorrência
1142                {
1143                    umidade = GeradorAleatorio.DistNormal(65.7125117, 5.5149343);
1144
1145                }
1146                if (enos == 1)// El Niño
1147                {
1148                    umidade = GeradorAleatorio.DistCauchy(201.365628, 18.631418);
1149
1150                }
1151
1152                if (enos == 2)// La Niña
1153                {
1154                    umidade = GeradorAleatorio.DistNormal(256.46, 35.201455);
1155
1156                }
1157
1158
1159            }
1160            if (mes == 12)//Dezembro
1161            {
1162                if (enos == 0) // Sem ocorrência
1163                {
1164                    umidade = GeradorAleatorio.DistNormal(63.7678461, 6.2982079);
1165
1166                }
1167                if (enos == 1)// El Niño
1168                {
1169                    umidade = GeradorAleatorio.DistCauchy(8.7017497, 1.0069021);
1170
1171                }
1172
1173                if (enos == 2)// La Niña
1174                {
1175                    umidade = GeradorAleatorio.DistWeibull(15.385826, 62.229428);
1176
1177                }
1178            }
1179
1180            float umidadef = Convert.ToSingle(umidade);
1181
1182            return umidadef;
1183        }
1184
1185    }
1186
1187
```

```
using System;

namespace GeradorValores
{
    /// <summary>
    /// Gerador de números aleatórios implementado para ser utilizado pelo jogo SIMCOW
    ///
    ///
    ///
    /// </summary>



    public class GeradorAleatorio
    {
        private static uint m_w;
        private static uint m_z;

        static GeradorAleatorio()
        {

            m_w = 0;
            m_z = 0;
        }

        public static void SetSemente(uint u, uint v)
        {
            if (u != 0)
                m_w = u;
            if (v != 0)
                m_z = v;
        }


        public static void SementeSystemTime()
        {
            System.DateTime datetime = System.DateTime.Now;
            long var = datetime.ToFileTime();
            SetSemente((uint)(var >> 16), (uint)(var % 4294967296));
        }

        // Produz uma amostra aleatória uniforme no intervalo aberto (0, 1).
        // Usa o algoritmo MWC de George Marsaglia para produzir um unsigned int.



        /// <summary>
        /// Método que retorna uma distribuição uniforme.
        /// </summary>
        /// <returns>Amostra Uniforme</returns>


        public static double DistUniforme()
        {

            SementeSystemTime();

            m_z = 36969 * (m_z & 65535) + (m_z >> 16);
            m_w = 18000 * (m_w & 65535) + (m_w >> 16);


            // 0 <= u < 2^32
            uint u = (m_z << 16) + m_w;
            // Número utilizado é 1/(2^32 + 2).
            // O resultado será entre 0 and 1.
            return (u + 1.0) * 2.328306435454494e-10;
        }




        /// <summary>
        /// Método que retorna uma distribuição normal.
```

```
74          /// </summary>
75          /// <returns>Obtem uma amostra aleatoria normal (Gaussiana) com média 0 e
            desvio padrão 1</returns>
76          public static double DistNormal()
77          {
78              // Algoritimo Box-Muller
79              double u1 = DistUniforme();
80              double u2 = DistUniforme();
81              double r = Math.Sqrt( -2.0*Math.Log(u1) );
82              double theta = 2.0*Math.PI*u2;
83              return r*Math.Sin(theta);
84          }
85
86
87
88          /// <summary>
89          /// Método que retorna uma distribuição normal com parâmetros espeficidados.
90          /// </summary>
91          ///  <param name="media">Média.</param>
92          /// <param name="desviop">Desvio Padrão.</param>
93          /// <returns>Obtem uma amostra aleatoria normal (Gaussiana) com média e
            desvio padrão especificados</returns>
94
95
96
97
98          public static double DistNormal(double media, double desviop)
99          {
100
101             return media + desviop * DistNormal();
102         }
103
104         /// <summary>
105         /// Método que retorna uma distribuição Gama.
106         /// </summary>
107         /// <param name="shape">Forma da distribuição.</param>
108         /// <param name="scale">Escala.</param>
109         /// <returns>Obtem uma amostra aleatoria Gama</returns>
110
111         public static double DistGama(double shape, double scale)
112         {
113             // Implementação baseada em "A Simple Method for Generating Gamma
                Variables"
114
115             double d, c, x, xquadrado, v, u;
116
117             if (shape >= 1.0)
118             {
119                 d = shape - 1.0/3.0;
120                 c = 1.0/Math.Sqrt(9.0*d);
121                 for (;;)
122                 {
123                     do
124                     {
125                         x = DistNormal();
126                         v = 1.0 + c*x;
127                     }
128                     while (v <= 0.0);
129                     v = Math.Pow(v, 3);
130                     u = DistUniforme();
131                     xquadrado = Math.Pow(x, 2);
132                     if (u < 1.0 -.0331* Math.Pow(xquadrado, 2) || Math.Log(u) < 0.5*
                        xquadrado + d*(1.0 - v + Math.Log(v)))
133                         return scale*d*v;
134                 }
135             }
136
137             else
138             {
139                 double g = DistGama(shape+1.0, 1.0);
140                 double w = DistUniforme();
141                 return scale*g*Math.Pow(w, 1.0/shape);
142             }
```

```
143             }
144
145
146             /// <summary>
147             /// Método que retorna uma distribuição de Weibull.
148             /// </summary>
149             ///  <param name="shape">Forma da distribuição.</param>
150             /// <param name="scale">Escala.</param>
151             /// <returns>Obtem uma amostra aleatoria de Weibull</returns>
152
153             public static double DistWeibull(double shape, double scale)
154             {
155                 if (shape <= 0.0 || scale <= 0.0)
156                 {
157                     string msg = string.Format("Shape e scale precisam ser positivos ",
                        shape, scale);
158                     throw new ArgumentOutOfRangeException(msg);
159                 }
160                 return scale * Math.Pow(-Math.Log(DistUniforme()), 1.0 / shape);
161             }
162
163
164             /// <summary>
165             /// Método que retorna uma distribuição Gama.
166             /// </summary>
167             ///  <param name="median">Média.</param>
168             /// <param name="scale">Escala.</param>
169             /// <returns>Obtem uma amostra aleatoria Gama</returns>
170
171             public static double DistCauchy(double median, double scale)
172             {
173                 if (scale <= 0)
174                 {
175                     string msg = string.Format("Scale precisa ser positiva", scale);
176                     throw new ArgumentException(msg);
177                 }
178
179                 double p = DistUniforme();
180
181                 // Aplica a inversa da distribuição de Cauchy em uma uniforme
182                 return median + scale*Math.Tan(Math.PI*(p - 0.5));
183             }
184
185
186
187             /// <summary>
188             /// Método que retorna uma distribuição Gama.
189             /// </summary>
190             /// <param name="mu">Média logaritimica.</param>
191             /// <param name="sigma">Desvio padrão logartitimico.</param>
192             /// <returns>Obtem uma amostra aleatoria Gama</returns>
193
194             public static double DistLogNormal(double mu, double sigma)
195             {
196                 return Math.Exp(DistNormal(mu, sigma));
197             }
198
199         }
200     }
201
```

GM.cs

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;


/// <summary>
/// Gerenciado do jogo
/// </summary>
/// <remarks>
/// Esse script é um singleton. Este padrão garante a existência de apenas uma instância de uma classe, mantendo um ponto global de acesso ao seu objeto.
/// Aqui é mantido todos os atributos que são de intereces de diversos objetos no jogo
/// </remarks>
public class GM : MonoBehaviour {  //Gerenciador do jogo



    //Declaração de inicialização do singleton
    public static GM instance = null;

    //Variáveis aqui são acessiveis de qualquer lugar!!
    //-------------------------------------------------
    public int velocidadeJogo = 1;

    public Propriedade propriedade;
    public Clima clima;

    public GameObject calendario;

    public Text temperatura, precipitacao, radiacao;

    //Lista de objetos Selecionados
    public List<SelectableUnitComponent> objSelecionados = new List<SelectableUnitComponent>();

    public GameObject gerenciadorMenu;

    public GameObject TelaGerencia;

    public GameObject menuAtivo;

    public GameObject cadeadoAberto;
    public GameObject cadeadoFechado;

    public GameObject sol;



    //public GameObject saveLoad;

    //-------------------------------------------------
    //-------------------------------------------------

    //inicialização do singletom
    void Awake(){
        if (instance == null)
            instance = this;
        else {
            if (instance != this)
                Destroy (gameObject);
        }

        //Inicialização variaveis
        propriedade = new Propriedade(100000);
        clima = new Clima (10, 0, 100,0,0,0);



        if (!GameObject.Find("SaveLoad").GetComponent<SaveLoadControl>().jogoSelecionado.Equals("novo")){
```

```
70              //Jogo carregado
71              string nome =
                GameObject.Find("SaveLoad").GetComponent<SaveLoadControl>().jogoSelecionad
                o;
72
                GameObject.Find("SaveLoad").GetComponent<SaveLoadControl>().Carregar(nome)
                ;
73              gerenciadorMenu.GetComponent<GerenciadorMenu>().LancarSucesso("Partida
                Carregada com sucesso!");
74          }
75          else {
76              //Novo jogo
77              GameObject.FindObjectOfType<CriarCenario>().criarChao();
78          }
79
80
81      }
82
83      void Start(){
84          //menuAtivo = null;
85          atualizarClima ();
86          AtualizarUI ();
87          cadeadoAberto.SetActive (true);
88          cadeadoFechado.SetActive (false);
89          clima.setEnos(0);
90      }
91
92      public void Descelecionar(){
93          objSelecionados.Clear();
94          objSelecionados = new List<SelectableUnitComponent>();
95      }
96
97      public void AtualizarUI(){
98          string texto = "";
99          texto += "Dados Climaticos\n";
100         texto += "++++++++++++++++++++++++" + "\n";
101
102         texto += "Temperatura mínima para o dia: " + clima.getTemperaturaMin() +
                "°C\n";
103         texto += "Temperatura máxima para o dia: " + clima.getTemperaturaMax() +
                "°C\n";
104         texto += "Temperatura atual: " + clima.getTemperatura() + "°C\n";
105         //texto += "Variador: " + clima.getVariadortemp() + "°C\n";
106
107
108         texto += "Luminosidade diária: " + clima.getLuminosidade() + "h\n";
109         texto += "Luminosidade Mensal: " + clima.getLuminosidadeMensal() + "h\n";
110
111         texto += "Umidade: " + clima.getUmidade() + "%\n";
112
113
114         texto += "Dias com precipitação: " + clima.getDiasPrecipitacao() + "\n";
115         texto += "Precipitação mensal: " + clima.getPrecipitacaoMensal() + "mm\n";
116         texto += "Precipitação acumulada: " + clima.getPrecipitacaoAcumulada() +
                "mm\n";
117         texto += "Precipitação diaria: " + clima.getPrecipitacaoDiaria() + "mm\n";
118         texto += "Precipitação momentânea: " + clima.getPrecipitacao() + "mm\n";
119
120
121         texto += "\n\n\nDados Propriedade\n";
122         texto += "++++++++++++++++++++++++" + "\n";
123         texto += "Caixa: " + propriedade.getCaixa().ToString("C0") + "\n";
124
125         texto += "\n\n\nEquipamentos\n";
126         texto += "++++++++++++++++++++++++" + "\n";
127         foreach(var equipamento in propriedade.equipamentos){
128             texto += equipamento.GetComponent<Equipamento>().nome + "\n";
129         }
130
131         texto += "\n\n\nProdutos\n";
132         texto += "++++++++++++++++++++++++" + "\n";
133         foreach(var produto in propriedade.produtos){
134             texto += produto.nome + " Quantidade: " + produto.quantidade + "\n";
```

```
135             }
136
137             texto += "\n\n\nAnimais\n";
138             texto += "++++++++++++++++++++++++" + "\n";
139             foreach (var animal in propriedade.animais)
140             {
141                 texto += "Raca: " + animal.raca + " Sexo: " + animal.sexo + " Peso: " +
                    animal.peso + "\n";
142             }
143             TelaGerencia.GetComponent<Text> ().text = texto;
144             //Debug.Log (texto);
145             gerenciadorMenu.GetComponent<GerenciadorMenu> ().atualizarUI ();
146             temperatura.text = clima.getTemperatura().ToString("N1") +"°C";
147             precipitacao.text = clima.getPrecipitacao().ToString("N1") +" mm";
148             radiacao.text = clima.getLuminosidade().ToString("N1") +" h";
149         }
150
151         public void novoDia(){
152             //arrumar a posicção do sol
153             sol.transform.rotation = Quaternion.Euler(-90f, 0f, 0f);
154             //atualizar o clima
155             atualizarTempMinMax();
156             atualizarPrecipitacaoDiaria();
157
158             AtualizarUI();
159         }
160
161         public void novaHora()
162         {
163             atualizarClima();
164             AtualizarUI();
165         }
166
167
168
169         public void novoMes(){
170             propriedade.pagar(ModuloFinanceiro.despesaMensal());
171             Debug.Log ("Despesa mensal: " + ModuloFinanceiro.despesaMensal());
172             AtualizarUI ();
173             clima.zerarPrecipitacaoAcumulada();
174             atualizarLuminosidadeMensal();
175             atualizarPrecipitacaoMensal();
176             atualizarTempMinMax();
177             atualizarPrecipitacaoDiaria();
178
179
180
181
182         }
183
184         public void novoAno(){
185             propriedade.pagar (ModuloFinanceiro.despesaAnual ());
186             Debug.Log ("Despesa Anual: " + ModuloFinanceiro.despesaAnual());
187             AtualizarUI ();
188             clima.setEnos(Random.Range(0, 2));
189         }
190
191         public void atualizarClima(){ //Atualização horária
192
193             float temperatura =
                ModuloClimatico.calcularTemperatura(clima.getTemperatura(),
                clima.getVariadortemp(), calendario.GetComponent<Calendario>().getHora());
194             clima.setTemperatura(temperatura);
195
196
197             float luminosidade =
                ModuloClimatico.calcularLuminosidade(clima.getLuminosidadeMensal(),
                calendario.GetComponent<Calendario>().getMes());
198             clima.setLuminosidade(luminosidade);
199
200             float precipitacao =
                ModuloClimatico.calcularPrecipitacao(clima.getPrecipitacaoDiaria(),
                clima.getPrecipitacaoMensal(), clima.getPrecipitacaoAcumulada());
```

```
201             clima.setPrecipitacao(precipitacao);
202             clima.adicionarPrecipitacaoAcumulada (precipitacao);
203 
204             float umidade =
                ModuloClimatico.calcularUmidade(calendario.GetComponent<Calendario>().getMes()
                , clima.getEnos());
205             clima.setUmidade(umidade);
206 
207         }
208 
209 
210 
211         public void atualizarTempMinMax() // Atualiza a temperatura mínima e máxima para
            o dia, e seta a temperatura inicial do dia
212         {
213                 float temperaturamin =
                    ModuloClimatico.calcularTemperaturaMin(calendario.GetComponent<Calendario>(
                    ).getMes(),clima.getEnos());
214             float temperaturamax =
                ModuloClimatico.calcularTemperaturaMax(calendario.GetComponent<Calendario>().g
                etMes(), clima.getEnos());
215 
216             clima.setTemperaturaMin(temperaturamin);
217                clima.setTemperaturaMax(temperaturamax);
218                float variadortemp = ModuloClimatico.calcularVariadorTemp(temperaturamin,
                   temperaturamax);
219                clima.setVariadorTemp(variadortemp);
220             clima.setTemperatura(temperaturamin + (6 * variadortemp));
221         }
222 
223 
224 
225             public void atualizarPrecipitacaoMensal()
226         {
227             int diasprecipitacao =
                ModuloClimatico.calcularDiasPrecipitacao(calendario.GetComponent<Calendario>()
                .getMes());
228             clima.setDiasPrecipitacao(diasprecipitacao);
229             float precipitacaomes =
                ModuloClimatico.calcularPrecipitacaoMensal(calendario.GetComponent<Calendario>
                ().getMes(),clima.getEnos());
230             clima.setPrecipitacaoMensal(precipitacaomes);
231         }
232 
233 
234 
235         public void atualizarLuminosidadeMensal()
236         {
237 
238             float insolacaomensal =
                ModuloClimatico.calcularLuminosidadeMensal(calendario.GetComponent<Calendario>
                ().getMes(), clima.getEnos());
239             clima.setLuminosidadeMensal(insolacaomensal);
240 
241         }
242 
243         public void atualizarPrecipitacaoDiaria()
244         {
245             int diasprecipitacao = clima.getDiasPrecipitacao();
246             float precipitacaodiaria =
                ModuloClimatico.calcularPrecipitacaoDiaria(clima.getDiasPrecipitacao(),clima.g
                etPrecipitacaoMensal(), clima.getPrecipitacaoAcumulada());
247             clima.setPrecipitacaoDiaria (precipitacaodiaria);
248 
249             if(precipitacaodiaria> 0)
250             {
251                 diasprecipitacao--;
252                 clima.setDiasPrecipitacao(diasprecipitacao);
253             }
254 
255             //clima.adicionarPrecipitacaoAcumulada(precipitacao);
256 
257            // int chovendo = UnityEngine.Random.Range(0, 1);
```

```
258
259         }
260
261             public void Salvar() {
262             GameObject.Find("SaveLoad").GetComponent<SaveLoadControl>().Salvar();
263         }
264
265     }
266
```

ANEXO A – GDD SIMCOW

SIMCOW

Documento de Projeto de Jogo (GDD)

Versão 2.0

Escrito por: Ricardo Peixoto Robaina

Revisado por: Ana Paula Lüdtke Ferreira

Data: 19/11/2018

Versão: 2.0

Classificação ESRB

Sumário

Histórico de revisão

Versão 1.0	09/06/2018
Versão 1.0	19/11/2018

Objetivos do Jogo

O objetivo deste jogo é o ensino de gestão de propriedades rurais de forma lúdica.

Visão Geral da História

O jogador tema responsabilidade de gerenciar uma propriedade rural pequena. Seu objetivo é aplicar técnicas de manejo na área disponível a fim de maximizar o lucro e assim expandir sua propriedade.

Regras do Jogo

O jogo não possui regras.

Condições de Vitória / Derrota

Não existem condições de vitória, pois não existe limite para o final de uma partida. A condição de derrota é caracterizada pela falência da propriedade rural.

Personagem do Jogador

O jogar possui um personagem anônimo que é o gestor da propridade rural.

Níveis de Jogo

O jogo conta com apenas um nível onde acontece a simulação de uma propriedade rural.

Métrica do Jogador

O sistema de métrica do jogo é baseno a realidade. Baseado nas unidades do SI. E sistema monetário em reais. A métrica é controlada pelo módulo externo métrica.

Pontuação

A pontuação do jogo é o dinheiro que a propriedade rual possui em caixa.

Música e Efeitos Especiais

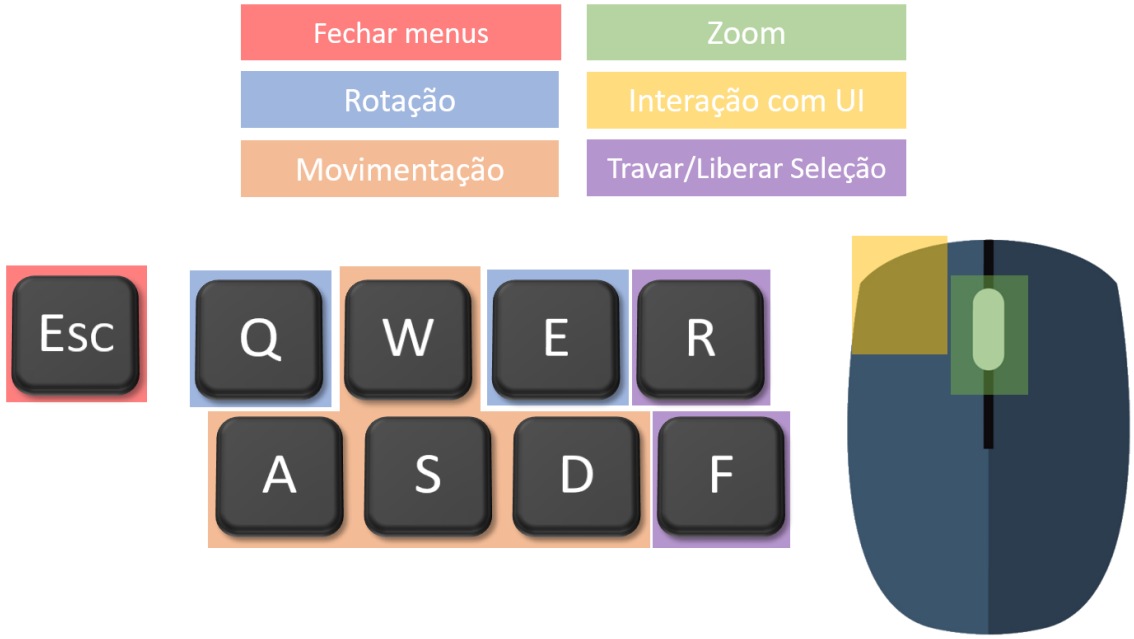
Uma música de ambiente foi adicionado na cena principal de jogo. Na versão atual, do jogo não foram utilizados efeitos sonoros.

Exigências de Tecnologia

- Jogo desenvolvido com a ferramenta Unity;
- Linguagem de programação utilizada é C#;
- Jogabilidade RTL com câmera Isométrica;
- Jogo desenvolvido para sistemas operacionais Windows, Linux e Mac Os;
- Sistama de salvamento do jogo executa operações de leitura e de escrita em um arquivo binário, alocado na memória secundária do computador.

Controles do Jogo

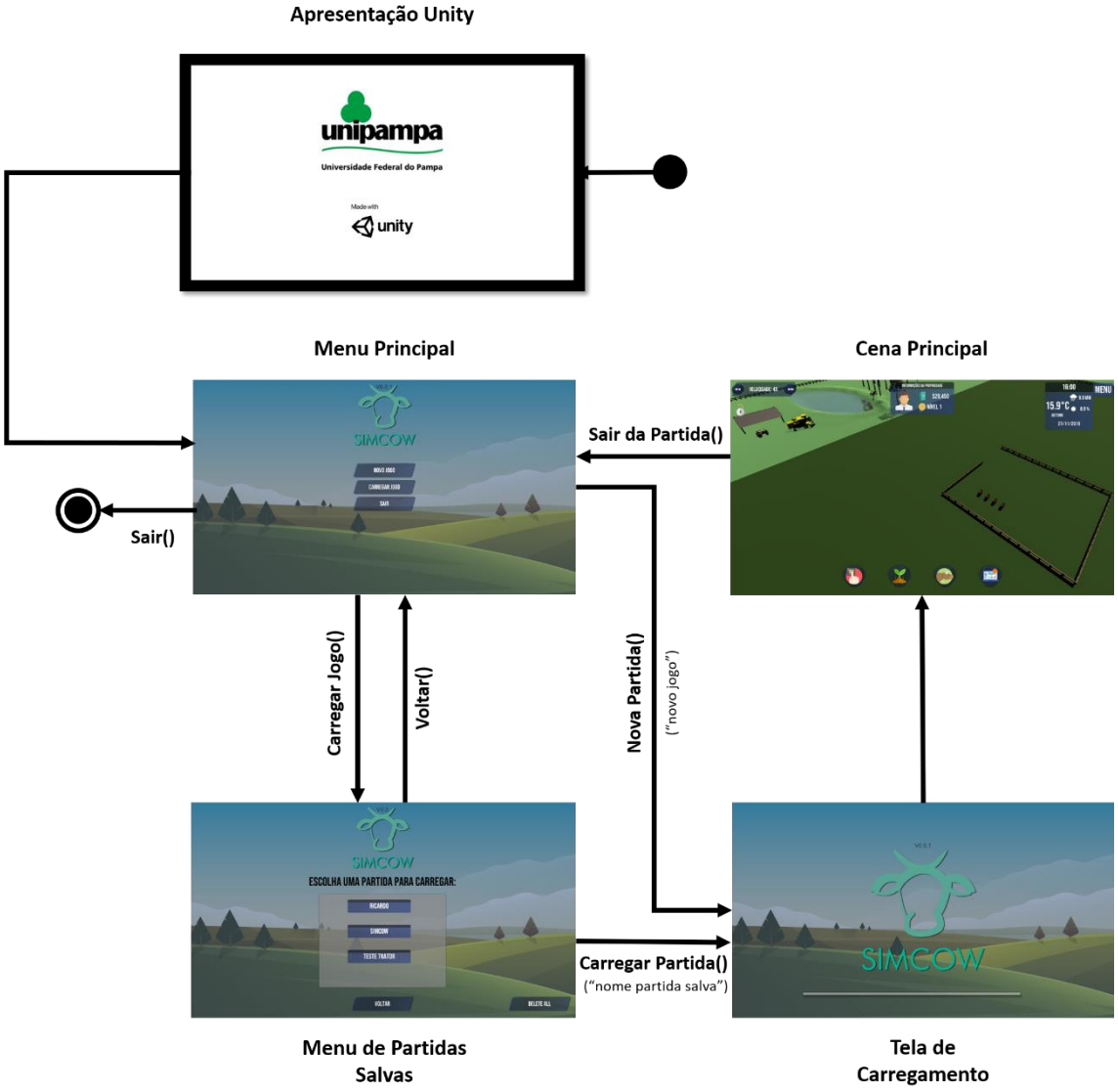
O jogador interage com o jogo através do sistema de HUD por meio dos seguintes controles.



Fluxo de Jogo

O jogo foi divido nas seguitnes cenas:

- **Apresentação Unity:** Cena com logotipos da Unity, da Unipampa e do curso de Engenharia de Computação.
- **Menu Principal:** Cena que contém as opções de inciar uma nova partida, de carregar uma partida salva e de sair do jogo.
- **Menu de Partidas Salvas:** Cena que contém as pardias salvas disponíveis para serem carredas.
- **Tela de Carregamento:** Apresenta o progresso do carregamento da *Cena Principal*.
- **Cena Principal:** Cena na qual a partida do jogo acontece.



Cena: Apresentação Unity

Cena: Menu Principal

Cena: Menu de Partidas Salvas

Cena: Tela de Carregamento

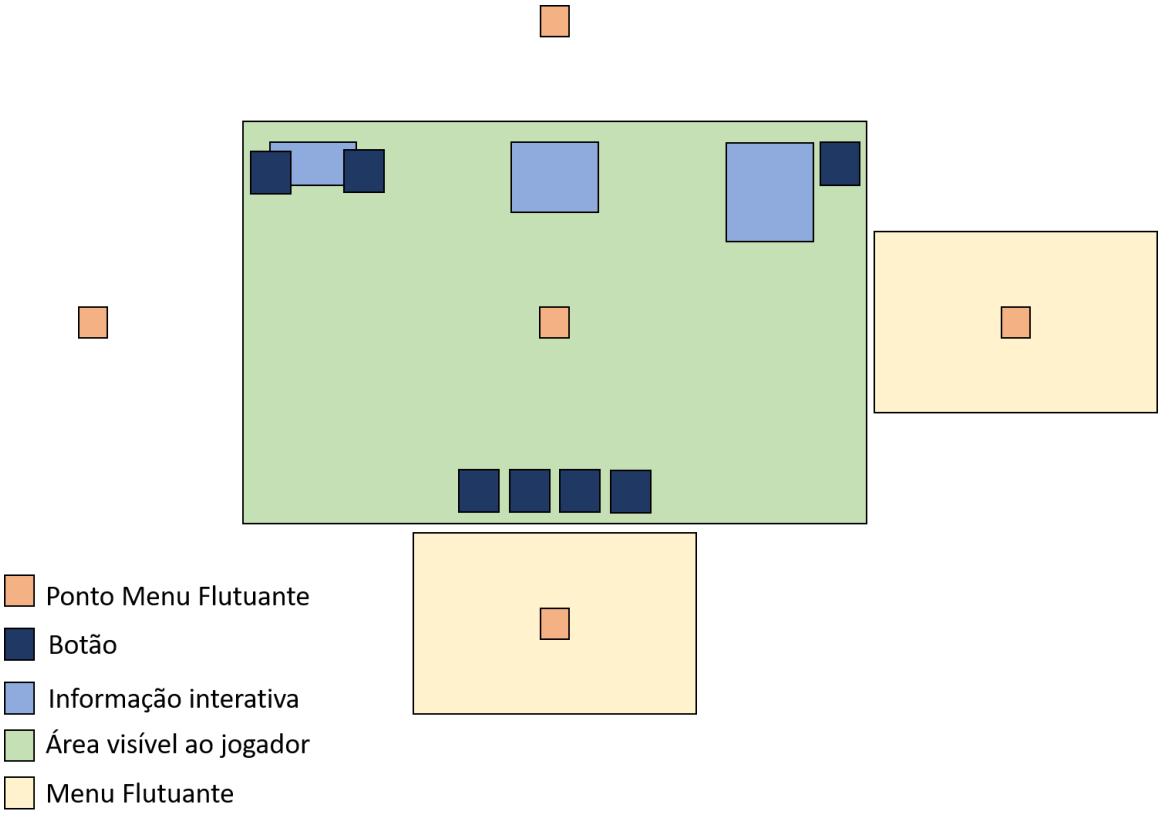
Cena: Principal

Câmeras de Jogo

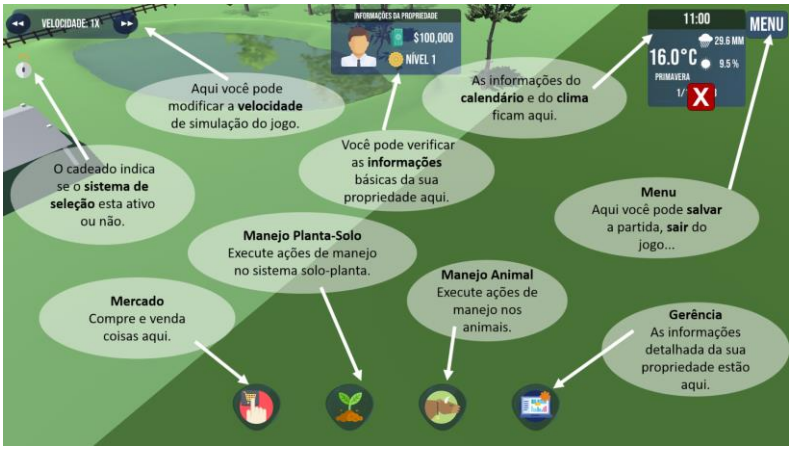
O jogo utiliza apenas uma câmera com visão isométrica.

Sistema de HUD

Os elementos de HUD são dispostos na periferia da tela. Estes elementos são dividios em elementos de saída, como os paineis de informação interativa e telas de alerta. Elementos de entrada como os botões. Além disso, os menus flutuantes agrupam elementos de entrada e de saída.

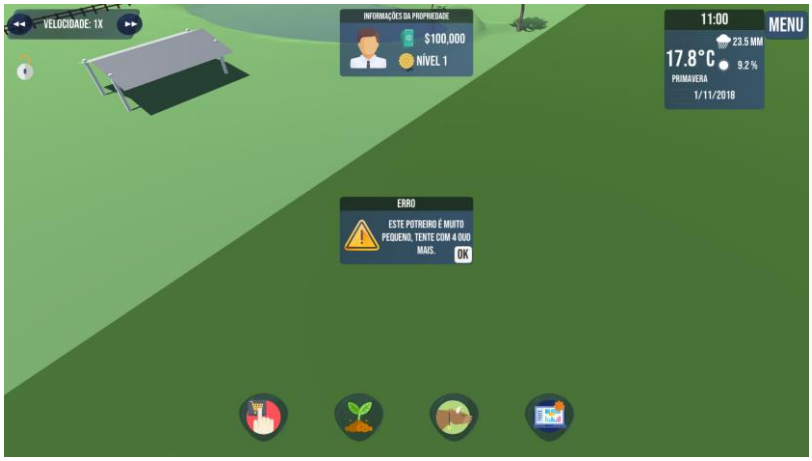


Elementos de HUD



Menu flutuante

Telas de alerta



Menus

Menu: mercado

Menu: manejo solo-planta

Menu: Manejo animal

Menu: Gerência

Menu: Pausa

Mecânicas do Jogador

- Modificar a velocidade de simulação
- Salvar a partida atual
- Apresentar a tela de controles e UI
- Sair do jogo
- Movimentar pelo cenário
- Selecionar áreas de interesse
- Travar e liberar o sistema de seleção
- Comprar equipamentos, trator, colheitadeira, plantadeira
- Comprar produtos, sementes e gado
- Vender produção vegetal
- Vender produção animal
- Técnicas de manejo no sistema solo-planta: arar, plantar, irrigar, adubar, colher
- Construir potreiros
- Migrar animais de potreiros
- Verificar informações de gerências

Edificações

- Potreiros

Equipamentos

- Trator
- Plantadeira
- Colheitadeira

Produtos

- Semente de azevém
- Semente de capim sudão
- Semente de soja
- Gado

Plantas

- Campo nativo
- Azevém
- Capim sudão
- Soja

Arquitetura

Diagrama de blocos

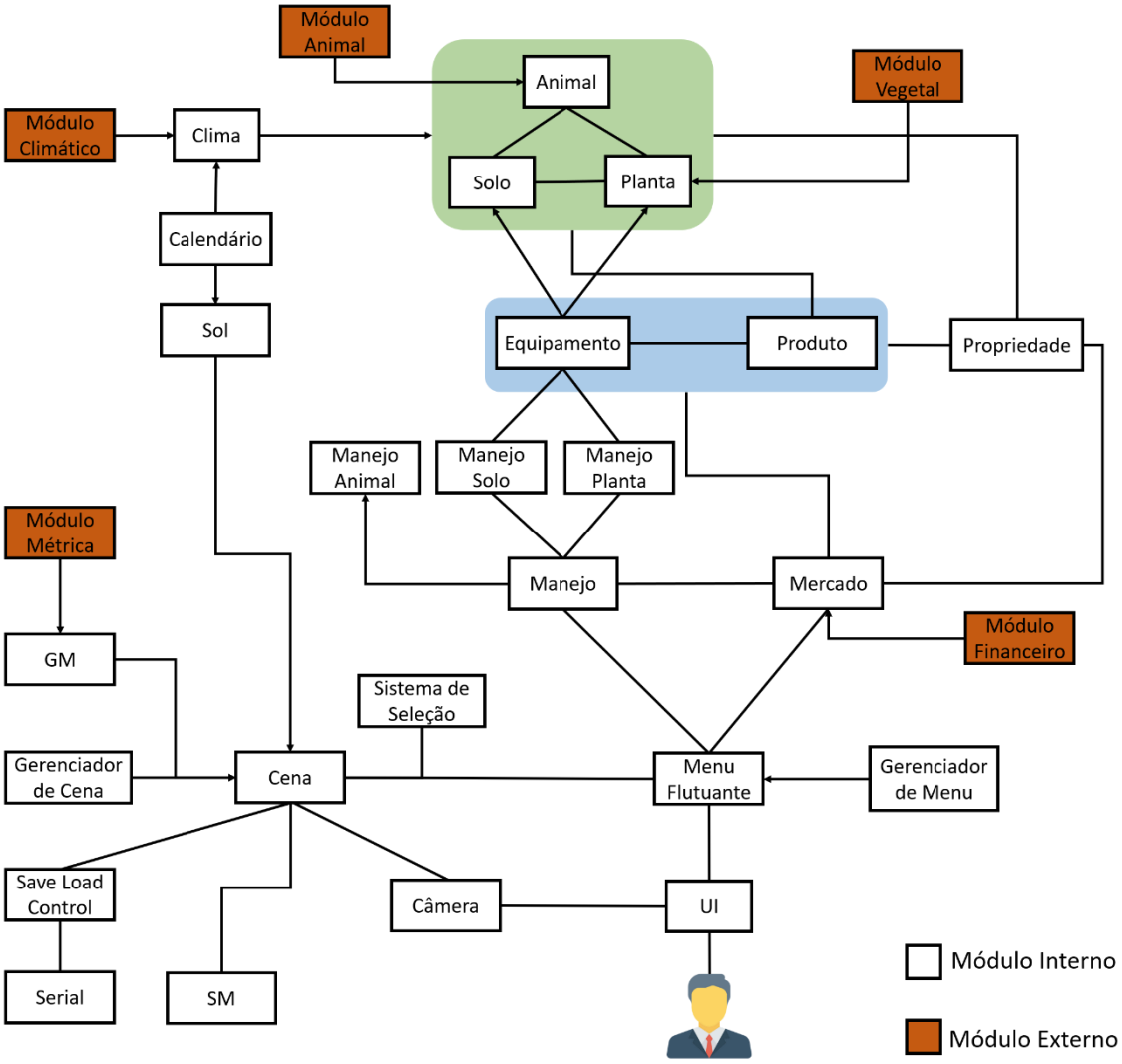


Diagrama de classes