

Instituto Federal de Educação, Ciência e Tecnologia do Ceará
Engenharia de Computação

Vitor Gonçalves de Aquino

**Ceará Game Tools: Uma ferramenta de software livre para geração
automática de games**

Fortaleza
2015

Instituto Federal de Educação, Ciência e Tecnologia do Ceará
Engenharia de Computação

Vitor Gonçalves de Aquino

Ceará Game Tools: Uma ferramenta de software livre para geração automática de games

Trabalho de Conclusão de Curso, apresentado à Banca Examinadora do Instituto Federal de Educação, Ciência e Tecnologia do Ceará para obtenção do grau de bacharel em Engenharia de Computação, sob a orientação do Prof. Dr. Carlos Hairon Ribeiro Gonçalves.

Fortaleza
2015

RESUMO: A crescente complexidade dos jogos eletrônicos tem alavancado a elaboração de ferramentas para o desenvolvimento de jogos. Entretanto, muitas dessas ferramentas exigem um conhecimento de programação. A criação de ferramentas que possibilitem a criação de jogos para usuários que não possuem conhecimento de programação possui um desafio muito grande. Baseado nisto, este artigo apresenta a criação da ferramenta Ceará Game Tools (CGT), um software de código de livre e multiplataforma para a criação de jogos em 2D de diversos gêneros que não necessita conhecimento de programação. A ferramenta possui um ambiente de desenvolvimento visual simples e intuitivo e uma biblioteca de imagens e sons disponibilizados para os usuários desfrutarem.

Palavras-chave: CGT; ferramenta;

ABSTRACT: The complexity of electronic games has been increased year by year. In order to help the game developers there are many software tools to create different types of games. However, many of these tools demand knowledge about programming. In this case create a game by simple users is practically not possible. In this context, in this article it will be discussed Ceará Game Tools software (CGT), free and multiplatform software made to enable users, without programming knowledge, creates 2D games. The tool has a simple and intuitive environment of development, images and sound library available for the users.

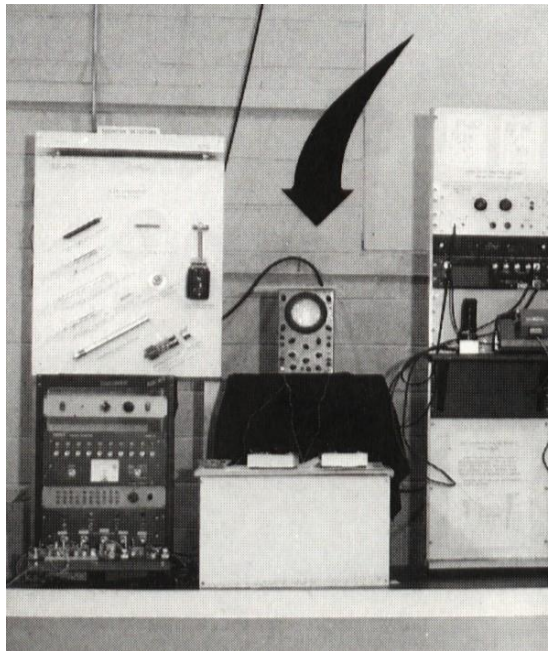
1 Introdução

1.1. História dos Jogos Eletrônicos

O surgimento dos jogos eletrônicos está diretamente ligado ao surgimento dos computadores, que se iniciou no período da segunda guerra mundial. A criação dos primeiros computadores como o ENIAC (*Electronic Numeral Integrator and Computer*) pelos americanos e a máquina de Turing pelo britânico Alan Turing, torna possível a criação dos jogos eletrônicos.

Em 1949, surge o embrião para os jogos eletrônicos, quando surge o projeto liderado por Ralph Baer, com o objetivo da criação da “melhor TV do mundo”. Este projeto visava uma participação interativa dos espectadores com o conteúdo que estivesse sendo criado na tela. Alan Turing em 1950 criou o “Teste de Turing”, um jogo onde existe um humano é o juiz e ele tem o objetivo de distinguir numa conversa, entre um humano e uma máquina, qual das pessoas é a máquina.

Figura 1 – Tennis for Two



Fonte: http://www.massarani.com.br/FGHQ_Tennisfortwo.html

Em 1958, o físico Willy Higinbotham criou o primeiro jogo eletrônico de que se tem ciência. Era um “jogo de tênis” realizado em um osciloscópio, sendo processado por um computador analógico que ficou conhecido como *Tennis for Two*.

Figura 2 - Spacewar



Fonte: <http://www.hardcoregaming101.net/spacewar/spacewar.htm>

Em 1962 três Stephen Slug Russel, Wayne Witanem e Martin Graetz criaram o *Spacewar* desenvolvido em Assembly. Um jogo de nave que surgiu com o objetivo de demonstrar todo o potencial de interatividade do computador.

A disseminação dos jogos eletrônicos aconteceu mesmo com a criação dos consoles na década de 70 que criava-se produtos especializados para o processamento de jogos, tornando-

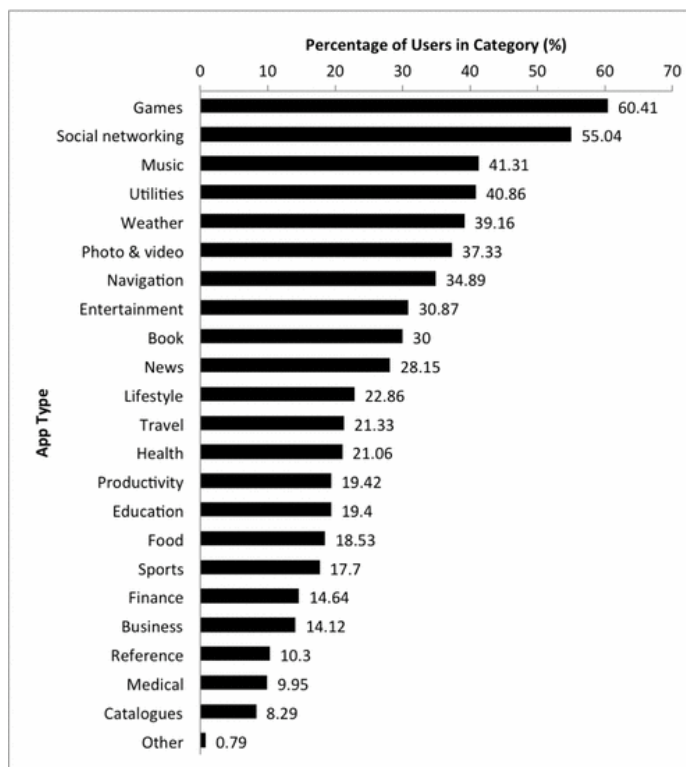
se mais barato e viável para a população adquirir. Consoles como Atari, Nintendo e Playstation bateram recordes de vendas, consolidando de vez o mercado de jogos eletrônicos. (Aranha, 2004).

1.2. Mercado de Jogos Eletrônicos

O mercado de jogos chegará a R\$ 93 bilhões de dólares em 2013, R\$ 14 bilhões dólares a mais que o ano anterior segundo a *Gartner*, Inc. Impulsionado pelo forte crescimento no mercado de jogos para dispositivos móveis. O mercado prever chegar a R\$ 111 milhões dólares em 2015.

Como o mercado de dispositivos móveis não para de crescer, o mercado de jogos para dispositivos móveis é o que mais cresce. A previsão é que dobre a receita nesse setor entre 2013 e 2015 chegando a R\$ 22 bilhões de dólares. Este crescimento vem pelo grande tempo que as pessoas estão utilizando os dispositivos móveis que são capazes de exibir jogos cada vez mais sofisticados (Stamford, 2013).

Figura 3 – Tipos de apps que usuários fazem download



Fonte: Lim *et al*, 2015

Como se pode ver no gráfico, o tipo de aplicativo mais utilizado pelos usuários são os relacionados a jogos com 60.4%. Isso evidencia o crescimento da área de jogos eletrônicos

voltados para dispositivos móveis, tornando uma ótima área para se investir e produzir conteúdo.

O mercado de jogos eletrônicos no Brasil já movimenta R\$ 900 milhões de reais. A indústria de videogame brasileira cresceu entre 9% e 15% nos últimos 15 anos. 61 milhões de brasileiros tem o costume de brincar com algum jogo eletrônico (Globo, 2015).

1.3. Identificação do Problema

Em termos culturais, o estado do Ceará é reconhecidamente um polo importante da cultura nordestina e se destaca por características religiosas, sertanejas, litorâneas e humorísticas. A divulgação dessa riqueza cultural em mídias eletrônicas não pode se limitar a *World Wide Web* (Grande Teia Mundial) da Internet. Os jogos eletrônicos, ou simplesmente, *games* são mídias de entretenimento que atende desde o público infantil até adultos. Com a massificação dos *smartphones*, os *games* casuais ganharam um novo canal de distribuição em nível mundial: lojas de aplicativos, tais como, *Google Play* e *App Store* da *Apple*.

Lojas deste tipo *e-commerce* permitem que qualquer produtor de mídia cultural (desenvolvedores de *games* e *softwares*, músicos, animadores, etc.) possa comercializar sua produção em um canal de nível mundial em que o retorno financeiro pode vir das vendas diretas dos artefatos culturais produzidos. Entretanto, a exploração comercial está além das vendas diretas supracitadas. Se um desses produtos atingir um público expressivo em termos de *marketing* nacional ou mesmo mundial, outras formas de geração de recursos poderão ser atingidas, por exemplo, o licenciamento de produtos que explorem a marca associada ao artefato cultural.

Esta monografia faz parte do projeto Ceará *Game Tools* (CGT) que tenciona desenvolver uma ferramenta de software livre para o desenvolvimento automático de *games* casuais em que qualquer pessoa com conhecimentos medianos de operação de microcomputadores poderá desenvolver *games*. Não haverá necessidade de codificação de *software* utilizando-se linguagens de programação, mas somente o desenvolvimento de modelos abstratos com o uso de aplicativos de *software* de uso livre. Inicialmente, o CGT contará com uma biblioteca de imagens, sons e animações relacionados com a cultura cearense. Entretanto, outras bibliotecas poderão ser desenvolvidas para abranger outros temas. O CGT deverá gerar *games* que possam ser executados nos sistemas operacionais iOS e Android. Desse modo, os desenvolvedores poderão aferir retorno financeiro com a venda e/ou popularização dos *games*, democratizando este canal para todas as pessoas que queiram ingressar nessa área.

1.4. Solução Estudada

Para resolver esse problema foi criado uma ferramenta de software livre, denominada Ceará *Game Tools* (CGT), que irá permitir a construção de games casuais sem a intervenção direta dos profissionais de desenvolvimento de software ou uso de linguagens de programação tradicionais. Qualquer pessoa com conhecimento mediano na operação de computadores estará apto para utilizá-lo. Essa arquitetura é composta por ferramentas de execução e configuração as quais possibilitam o desenvolvimento sem necessidade de escrita de código fonte.

1.5. Disposição do Documento

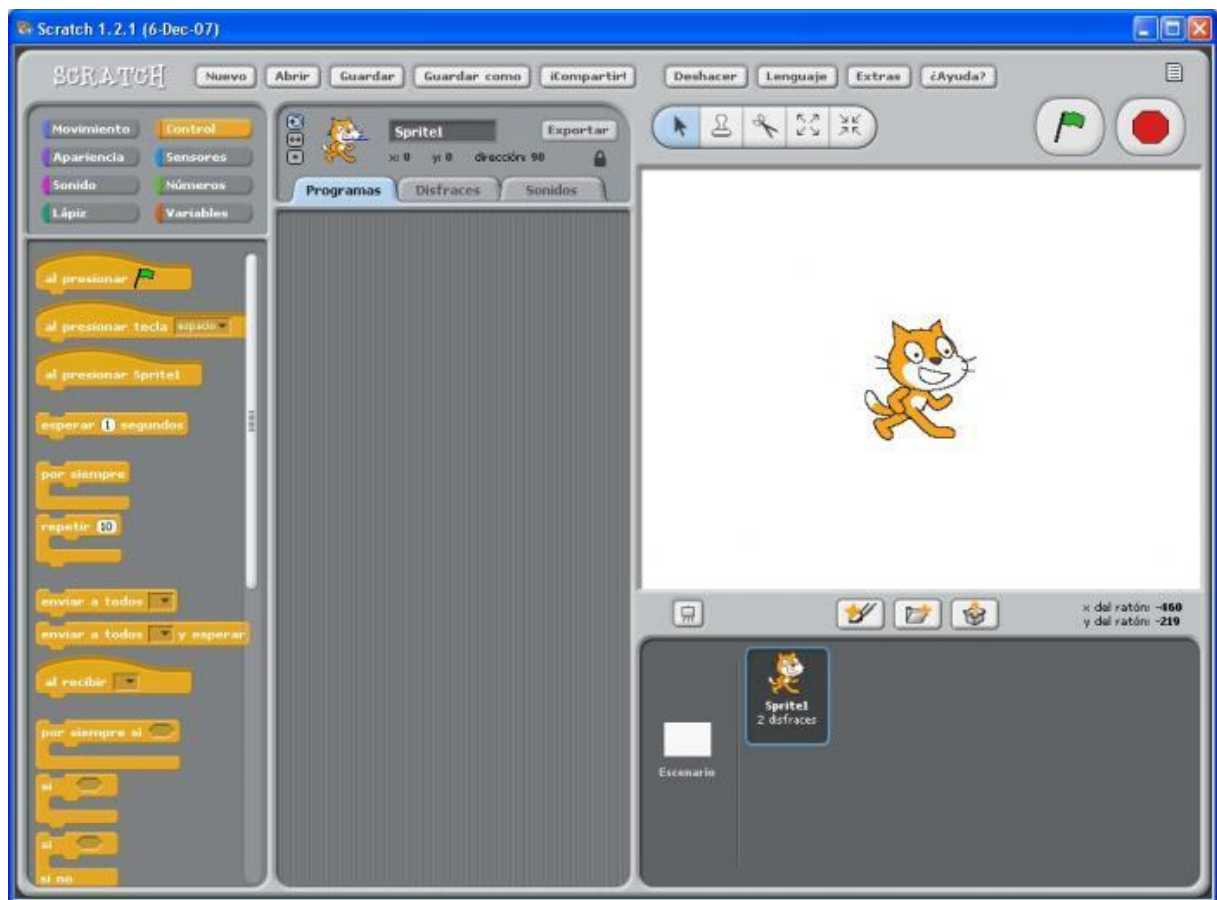
Na seção 1.0 é descrita a introdução, logo após na seção 2.0 é relatado os trabalhos relacionados com esse documento. A seção 3.0 é descrito métodos e ferramentas para o desenvolvimento do software. A seção 4.0 é descrita a forma com que o software foi desenvolvido e na 5.0 é concluído com os benefícios causados e trabalhos futuros.

2. Trabalhos Relacionados

Para atrair o público que não possui conhecimento em programação, foram criadas algumas ferramentas. Elas permitem a criação de jogos por meio da montagem do cenário somente arrastando objetos e definição de comportamento através de menus.

2.1. Scratch

Figura 4 – Ambiente de Desenvolvimento Scratch



Fonte: <http://portable-scratch.softonic.com.br/>

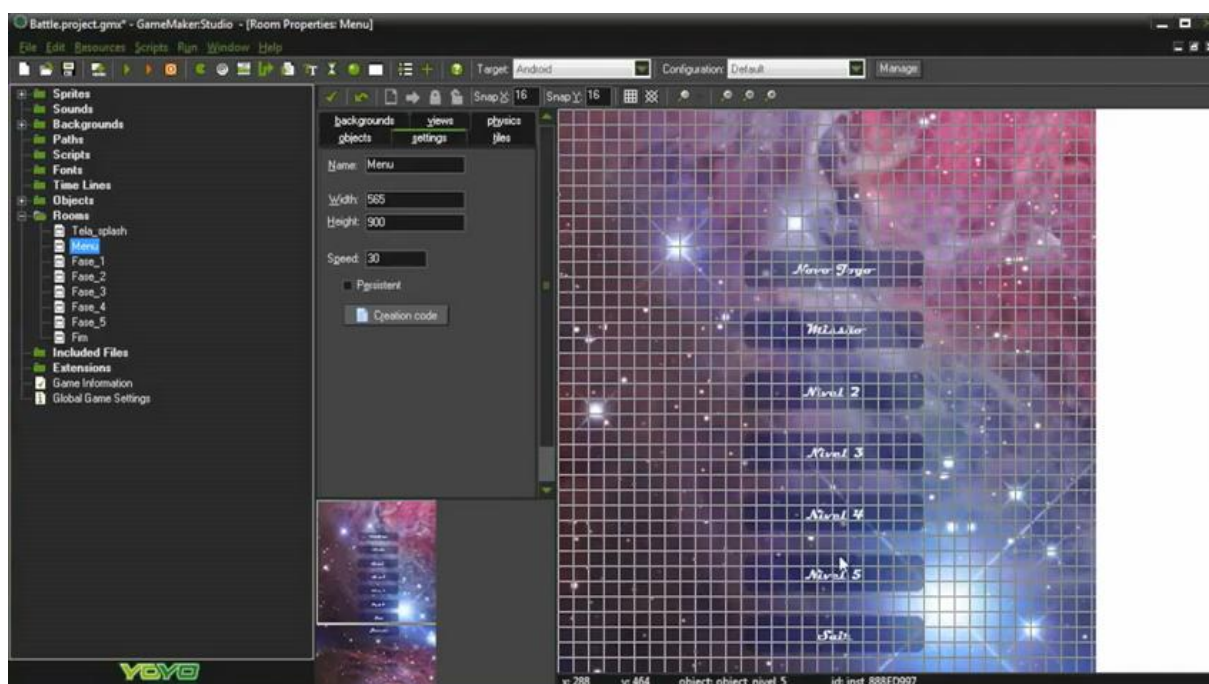
Scratch é uma linguagem de programação desenvolvida no Media Laboratory do Massachusetts Institute of Technology (MIT), baseado na linguagem de programação LOGO por Seymour Papert. É bastante simples e intuitiva baseada no “clique e arrastar”. Foi criada voltada para o público infantil, oferecendo múltiplas opções de línguas o que facilita na utilização.

Mesmo sendo uma linguagem o processo de aprendizagem é muito rápido. A sintaxe não é rígida para que se tenha atenção somente na lógica do que você está fazendo. É

disponibilizada a criação customizada de personagens, cenários e áudios (Andrade *et al*, 2013).

2.2. Game Maker Studio

Figura 5 – Ambiente de Desenvolvimento Game Studio Maker



Fonte: <http://www.superdownloads.com.br/download/178/gamemaker-studio/>

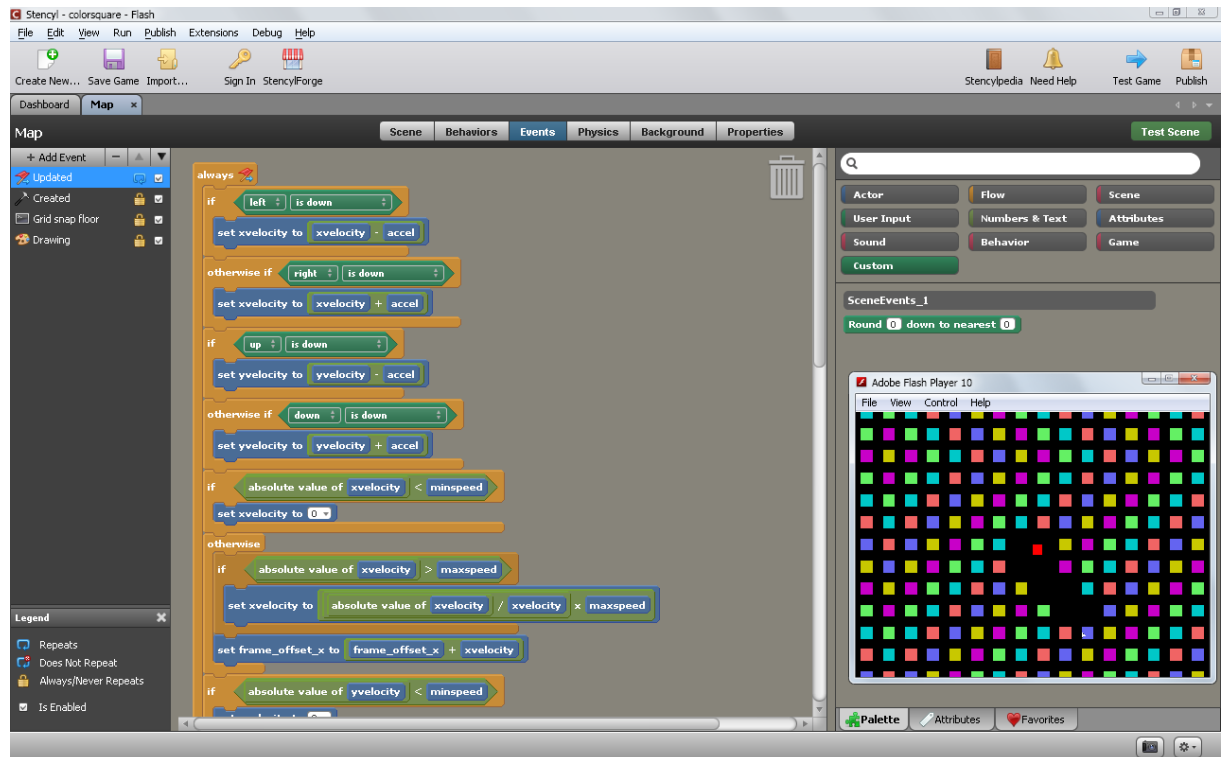
Game Maker Studio é um motor de desenvolvimento de jogos 2D que atende tanto os iniciantes em desenvolvimento de jogos, quanto os profissionais. O *Game Maker* foi criado por Mark Overmars, um cientista da computação e professor de jogos da universidade de Utrecht.

Esse motor possui a possibilidade de fazer jogos sem o conhecimento de programação, mas possui um ambiente de desenvolvimento próprio e uma linguagem própria para pessoas que querem fazer algo mais complexo. Isso torna o motor mais flexível. É uma ferramenta que possui uma versão grátis, mas com poucas funcionalidades liberadas.

O *Game Maker Studio* permite a criação de jogos sem se preocupar com a plataforma que irá ser escolhida. Ele pode gerar jogos para: iOS, Android, Windows Phone e HTML5, e nas plataformas fixas: Windows, Mac OS X e Ubuntu (Brito e Carvalho, 2013).

2.3. Stencyl

Figura 6 – Ambiente de Desenvolvimento do Stencyl

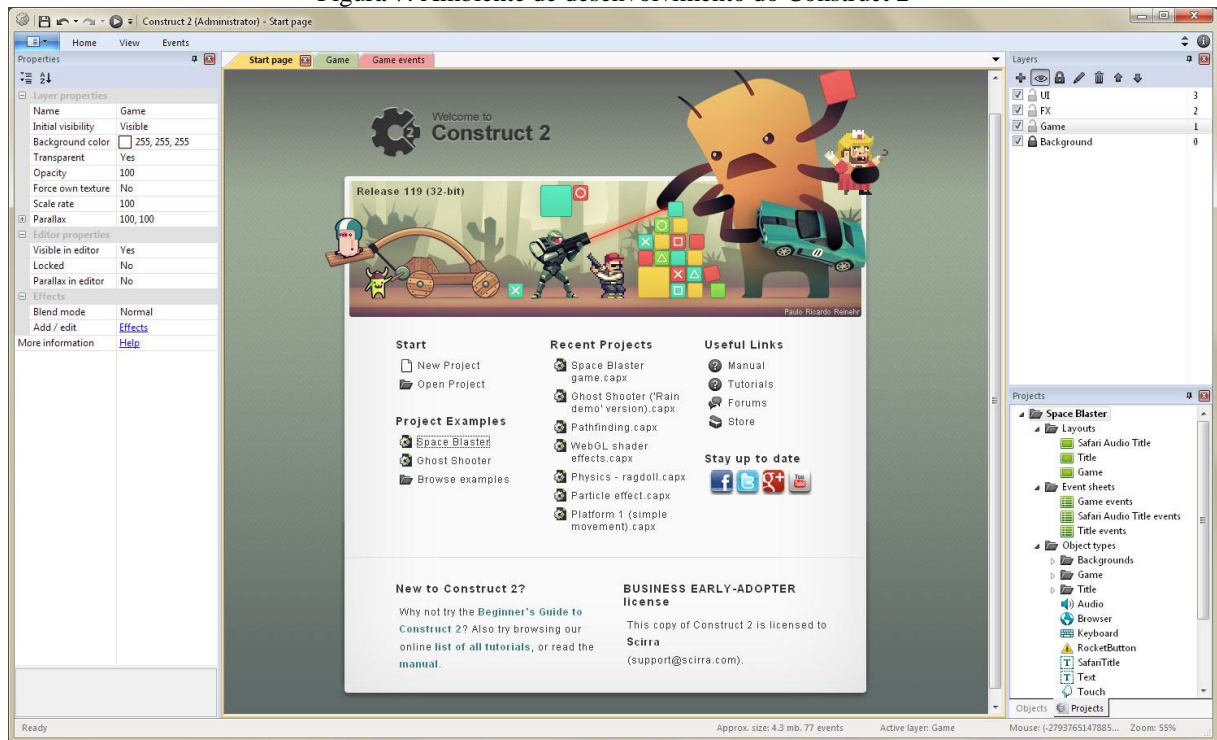


Fonte: <http://brendanzagaeski.appspot.com/0010.html>

Stencyl é um motor de jogos desenvolvido por Jonathan Chung em 2011. É uma ferramenta para criação de jogos 2D com ambiente gráfico e que permite a programação através de blocos, o que torna bem mais fácil para pessoas que não possuem conhecimentos em programação. É uma ferramenta que possui uma versão gratuita e versões pagas que permite exportar para dispositivos móveis, desktop (Stencyl, 2015).

2.4. Construct 2

Figura 7: Ambiente de desenvolvimento do Construct 2



Fonte: <https://www.scirra.com/construct2>

Construct 2 é um motor de jogos desenvolvido pela empresa Scirra Ltda. O público alvo são pessoas que não sabem programar, permitindo a criação de jogos rápidos, por meio do estilo *drag-and-drop*. É um motor de jogos 2D baseado em *HTML5* com uma boa documentação e interface intuitiva. Possui uma versão grátis para *download* e um suporte para os usuários. Possui exportação tanto para *desktops* quanto para *mobile* (Scirra, 2015).

A tabela 1.1 mostra um comparativo da ferramenta CGT com os outros motores de jogos citados. A comparação foi feita com a versão grátis de cada um dos motores.

	CGT	Scratch	Game Maker	Stencyl	Construct 2
Possui versão grátis?	Sim	Sim	Sim	Sim	Sim
Possui Versão para Windows, iOS e Linux?	Sim	Não	Sim	Sim	Sim
Exporta para Android e iOS?	Sim	Não	Não	Não	Não
Precisa programar?	Não	Sim	Não	Não	Não
Possui Código Aberto?	Sim	Não	Não	Não	Não
Ambiente de Pré Visualização?	Não	Não	Sim	Sim	Sim

3. Métodos e Ferramentas

Para o desenvolvimento da aplicação foram utilizadas várias ferramentas. A aplicação foi desenvolvida para funcionar em *desktops* que possuam o Java 8 instalado. O software possui como uma das camadas a libGDX que é uma biblioteca para criação de jogos gratuita (BrownleeJ. 2013). Foi utilizada o Gradle uma ferramenta de build e controle de dependências (Dockter e Murdock, 2007) e também Git um sistema de controle de versão para auxiliar no desenvolvimento (Chacon e Straub, 2014). Todo o projeto foi desenvolvido baseado na metodologia ágil SCRUM (Schwaber, 2009).

3.1. Plataforma Java

Java é uma linguagem de programação e uma plataforma de computação lançada pela Sun Microsystems, em 1995. É a tecnologia que capacita muitos programas, como utilitários, jogos e aplicativos corporativos, entre outros. O Java é executado em mais de 850 milhões de computadores pessoais e em bilhões de dispositivos em todo o mundo, inclusive telefones celulares e dispositivos de televisão. (ORACLE, 2013).

3.2. LibGDX

LibGDX é um *framework* de desenvolvimento de jogos em Java criado em 2010 que permite você desenvolver seu jogo e executá-lo em várias plataformas. Existe uma abstração entre as diferentes plataformas e é fornecido um sistema modular unificado para arquivo I/O, periféricos de entrada, gráficos, áudio e gerenciamento de ciclo de vidas e muitas outras ferramentas para auxiliar no desenvolvimento. Possui integração com várias bibliotecas de terceiros como *box2d* e *freetype*, uma boa documentação e uma comunidade ativa (Brownlee. 2013).

3.3. Gradle

Gradle é uma ferramenta de automação de build, testes, publicação e *deploy* de *software*. Além disso ele realiza o gerenciamento de dependência da aplicação. Ele tem por objetivo baixar as dependências do projeto (Dockter e Murdock, 2007).

3.4. Git

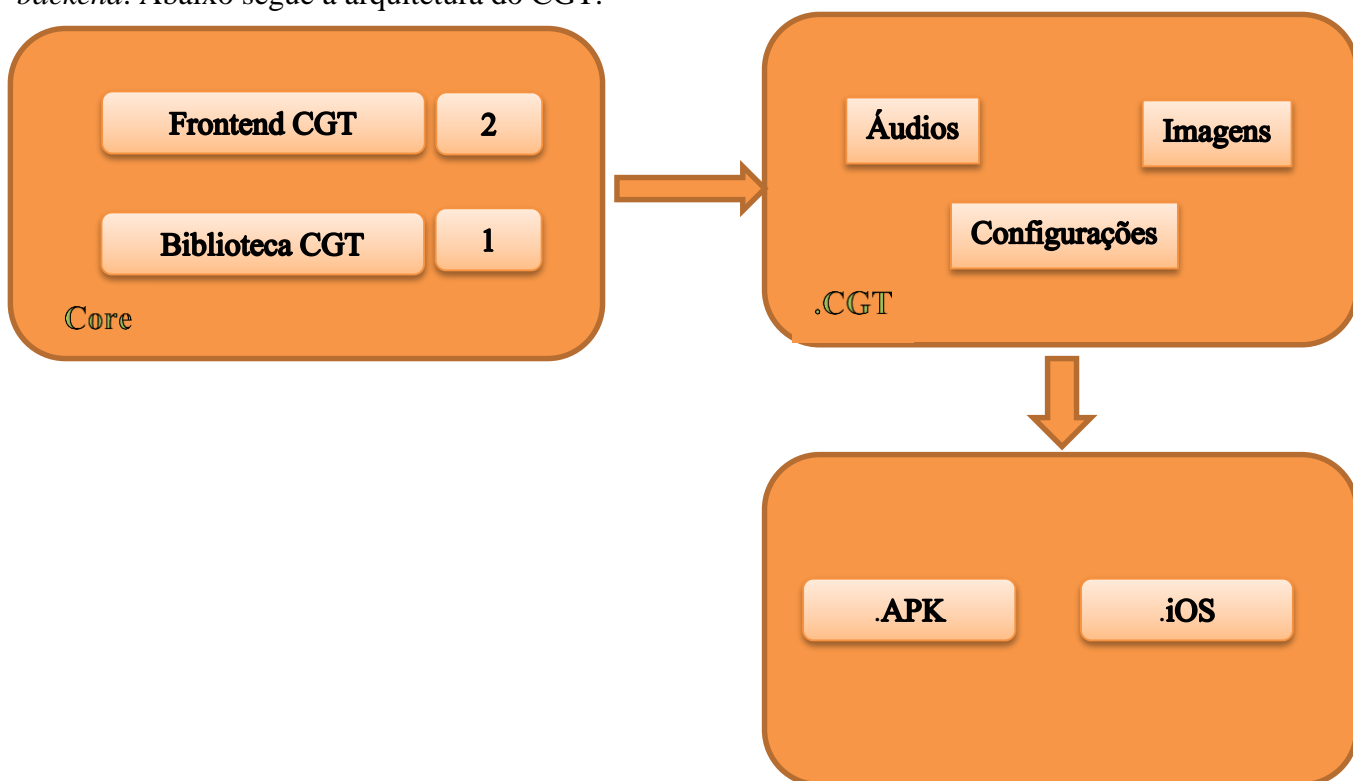
Git é um sistema de controle de versão distribuído criado por Linus Torvalds. Tem por objetivo auxiliar no desenvolvimento de software quando se trabalha em equipe. Ele possui ênfase na velocidade e é software livre (Chacon e Straub, 2014).

3.5. Scrum

Scrum é um metodologia ágil que baseia o desenvolvimento de software focando na flexibilidade, adaptabilidade e produtividade. As características do scrum são: equipes pequenas, requisitos pouco estáveis, interações curtas para possibilitar a visualização do desenvolvimento. É dividido em *sprints* onde se define as funcionalidades que serão desenvolvidas e reuniões diárias curtas para se atualizar do que está acontecendo no projeto (Schwaber, 2009).

4. Descrição do Software

O CGT é dividido em dois módulos: o *backend* implementado em Java e que possui toda a estrutura de classes do CGT e o *frontend* implementado com JavaFX que interage com o *backend*. Abaixo segue a arquitetura do CGT.



A seguir, uma descrição do *backend* que foi organizado em pacotes:

- **Pacote *policy***

Por ser um jogo totalmente configurável, a presença de políticas irá facilitar o processo de programação. As políticas são *enumeration* java e definem, por exemplo, os possíveis eventos de entrada do usuário, os diversos comportamentos, ações de movimentações, tipos de vitória, de derrota. Um exemplo de política é a de movimentação do ator que pode ser em dois sentidos, quatro sentidos ou oito sentidos.

- **Pacote *unit***

O pacote *unit* apresenta todas as classes unitárias do Ceará Game *Tools*. Desta forma, os objetos que serão únicos para todo o projeto, ou objetos que identificam outros objetos no projeto *game*, estão presentes neste pacote.

- **Pacote *core***

O pacote *core* é o mais importante de todo o projeto. Nele estão presentes as classes responsáveis pela modelagem dos objetos do jogo. Um jogo pode possuir ator, inimigos, opositores, bônus e projétil.

- **Pacote *behaviors***

Todos os objetos do jogo possuem um comportamento. Este comportamento está presente no pacote *behaviors*, através de políticas criada no pacote de políticas. Neste pacote contém as implementações das políticas de movimentação, como por exemplo, a implementação da política dois pontos, onde o inimigo vai ficar andando entre dois pontos fornecidos.

- **Pacote *cgt***

Neste pacote estão presentes as classes responsáveis por modelar a lógica e dinâmica jogo. Um jogo possui os objetos que foram modelados no pacote *core*, assim como um conjunto de telas. Este pacote contém a classe *game* que contém uma lista de

mundos que o seu jogo possui, junto com as telas de menus e configurações que foram criadas.

- **Pacote util**

O pacote util contém um conjunto de classes auxiliares que são utilizadas pelas outras classes dos outros pacotes. Neste pacote existem classes como File que auxilia na leitura de arquivos, Animation responsável pela animação dos objetos.

- **Pacote win**

O pacote *win* é responsável pela implementação de todas as políticas de vitória que o seu jogo pode possuir. Uma política de vitória pode ser o ator destruir todos os inimigos, pegar todos os bônus, sobreviver durante um determinado tempo.

- **Pacote lose**

O pacote *Lose* é responsável por implementação de todas as políticas de derrota que o seu jogo pode possuir. Um exemplo de política de derrota seria o ator morrer.

O módulo de *frontend* foi bastante abordado no anexo 2 (manual do usuário), onde é caracterizado todas as telas que o CGT possui, assim como a configurações de todos os objetos que o seu jogo pode conter.

5. Conclusão

Nesse trabalho apresentou-se uma ferramenta para criação de jogos 2D que tem por objetivo tornar muito simples o processo de criação de jogos. Desta forma, o usuário com nenhum conhecimento em programação poderá construir seus próprios jogos e disponibilizá-los em lojas de aplicativos como: *Google Play* e *App Store* da *Apple*.

Foi feito uma Game Jam, um evento onde pessoas se reúnem para desenvolver jogos normalmente durante um fim de semana, com a participação 10 pessoas voluntárias que não possuíam conhecimento na área de jogos e que tinham por objetivo produzir jogos temáticos. Um dos temas foram jogos voltados para área de educação, onde foram criados dois jogos. O primeiro, foi um jogo de *quiz* de conhecimentos gerais, o segundo foi voltado para calouros que ingressam no Instituto Federal do Ceará para ensinar onde ficam os locais básicos do campus e funções, como: biblioteca, Departamentos e outros. Mais três jogos foram

produzidos com o tema livre, totalizando ao todo cinco jogos. Nas figuras abaixo, seguem as imagens dos jogos produzidos.



Figura 8 – Jogo 1

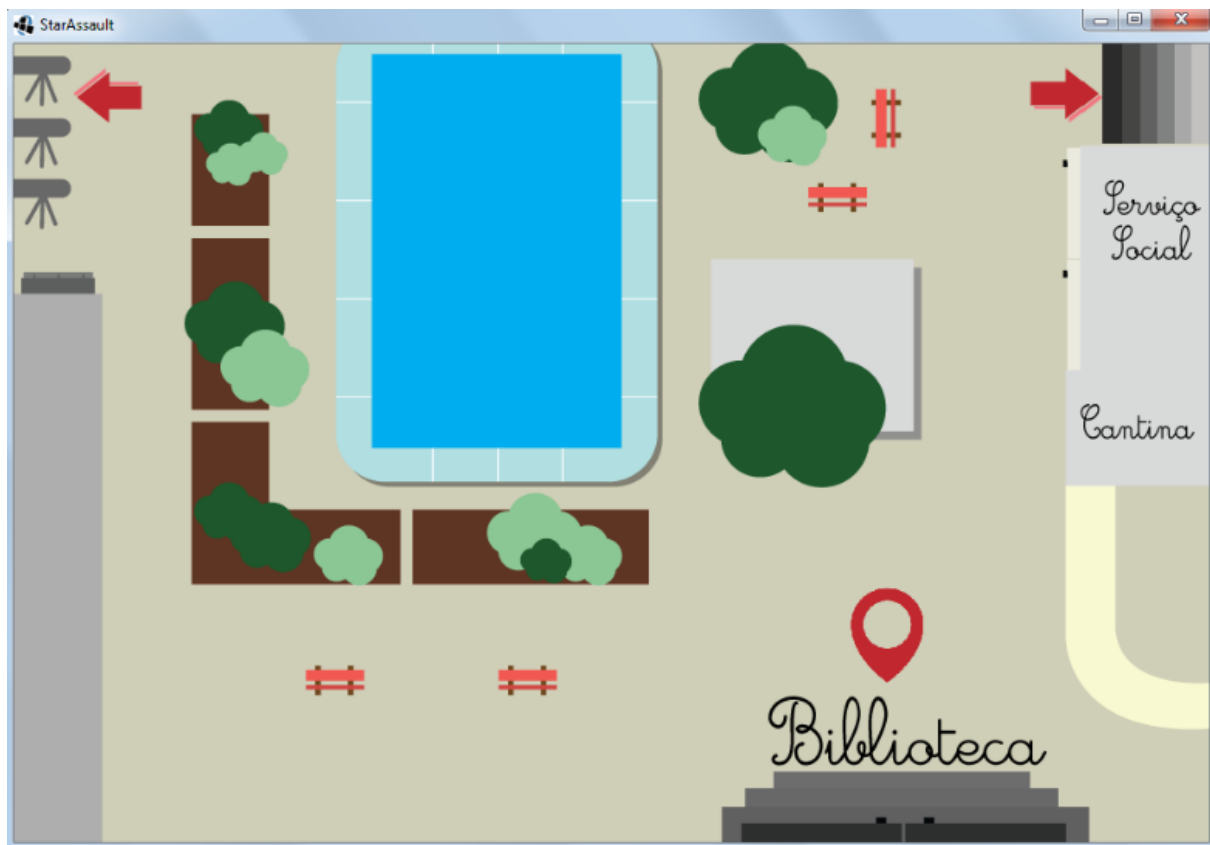


Figura 9 – Jogo 2.

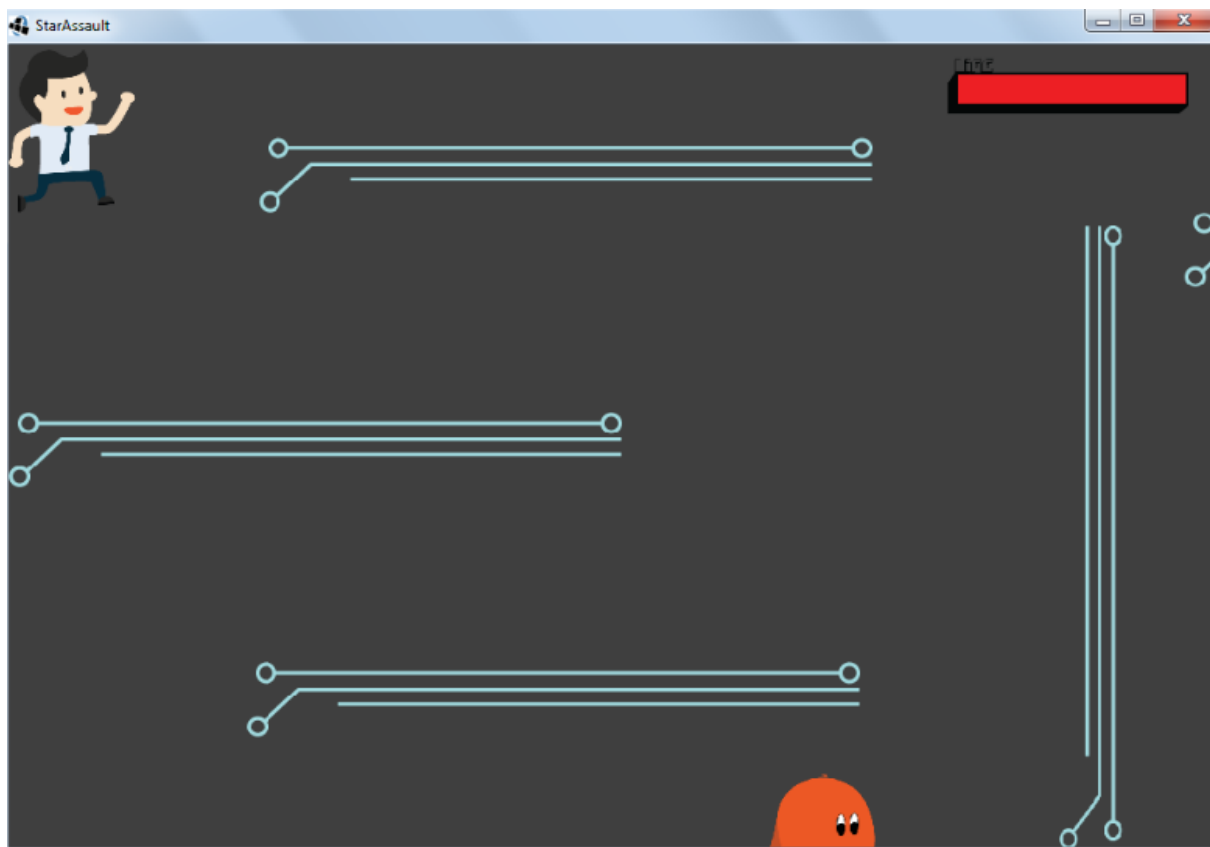


Figura 10 – Jogo 3.

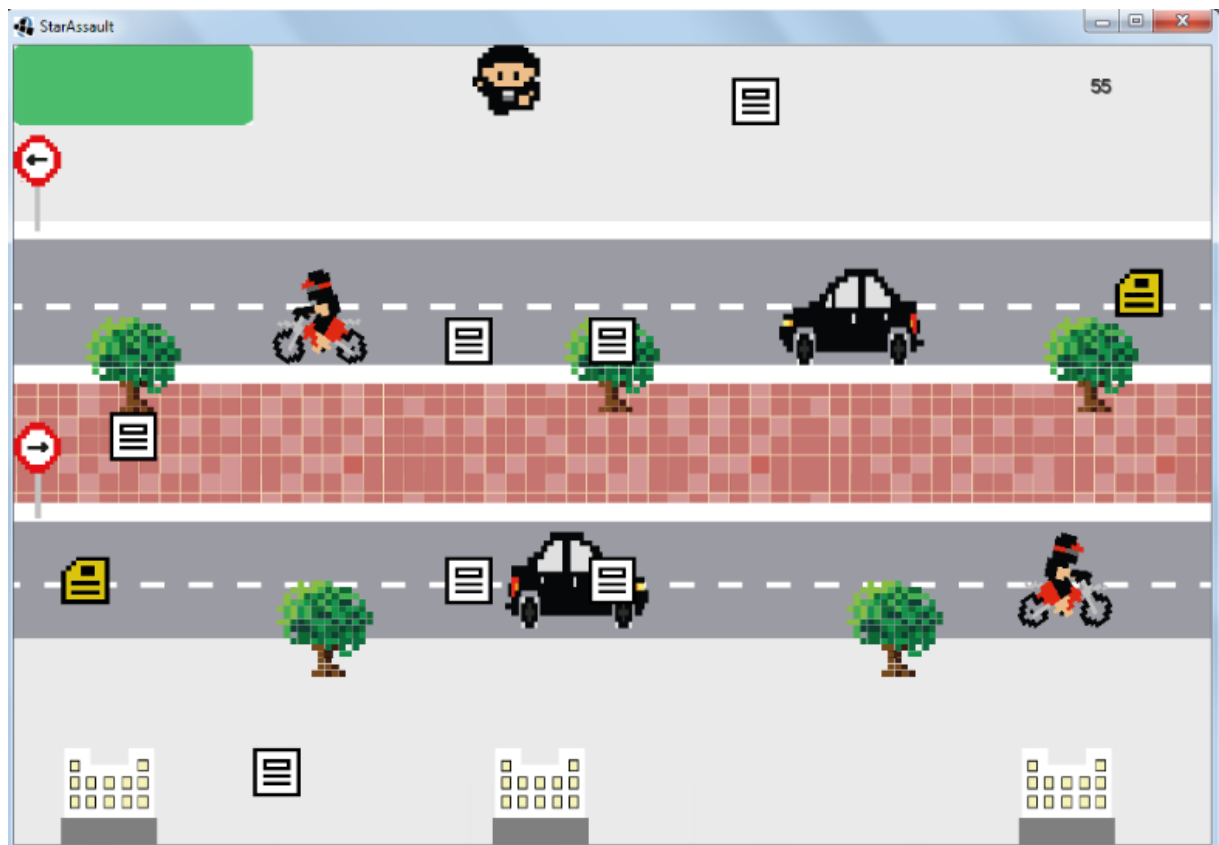


Figura 11 – Jogo 4.

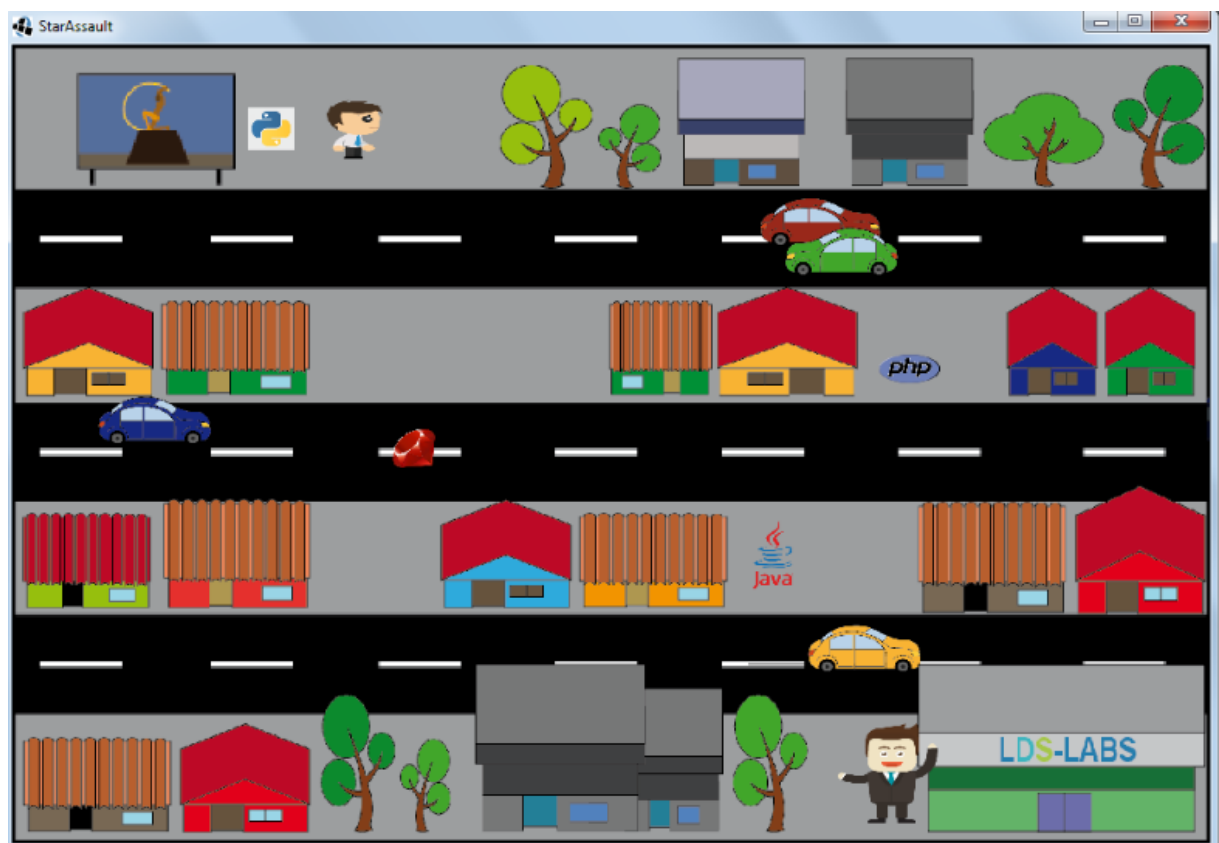


Figura 12 – Jogo 5.

Para trabalhos futuros seria interessante a criação de novas políticas para aumentar as opções dos jogos, implementar novas estruturas de jogos, já que existem várias estruturas e o CGT atende uma parte delas, como por exemplo, jogo do tipo *runner*, jogos sem personagem principal, aumentar o número de plataformas que o jogo pode ser exportado, como por exemplo, *windows phone*, *WEB*, *desktop*., adicionar módulo de pré-visualização do jogo.

Referências

Andrade, M; Silva, C; Oliveira, T. Desenvolvendo games e aprendendo matemática utilizando o Scratch. 2013.

Aranha, G. O processo de consolidação dos jogos eletrônicos como instrumento de comunicação e de construção de conhecimento. 2004.

Brito, WMA; Carvalho BO. Processo de Desenvolvimento de um Jogo utilizando Game Maker Studio. 2013

Brownlee, J. Mobile Game Engine Interviews with Mobile Game Engine Developers. 2013

Chacon, S;Straub, B. Pro Git: everything to know about git. Mountain View. 2014

Dockter, H; Murdock, A. Gradle User Guide. 2007

Globo, Mercado de games movimenta R\$ 44 milhões e deve crescer em 2015, Disponível em: <<http://g1.globo.com/pernambuco/noticia/2015/02/mercado-de-games-movimenta-r-44-mi-em-pe-e-quer-crescer-em-2015.html>> Acesso em 7 de março de 2015

ORACLE, O que é a tecnologia Java e porque é necessária? . Disponível em: <http://www.java.com/pt_BR/download/faq/whatis_java.xml>. Acesso em 7 de março de 2015.

Schwaber, K. Guia do Scrum. Scrum alliance. 2009

Scirra, Disponível em: <<https://www.scirra.com/construct2>> Acesso em 14 de março de 2015

Stamford, Gartner Says Worldwide Video Game Market to Total \$93 Billion in 2013, Disponível em: <<http://www.gartner.com/newsroom/id/2614915>> Acesso em 7 de março de 2015

Stencyl, Disponível em: <<http://www.stencyl.com/>> Acesso em 14 de março de 2015

Anexo 1

Requisitos Funcionais

Esta parte do documento traz o preciso detalhamento dos requisitos funcionais do sistema, ou seja, aqueles que são imprescindíveis para o seu pleno funcionamento.

Aplicação Mobile

[RF1.01] Gerar uma versão para dispositivos móveis Android.

A aplicação deve funcionar em dispositivos móveis com sistema Android.

Atores: Usuário

Prioridade: Essencial

Entradas e pré-condições: Usuário deve criar um novo projeto, configura-lo e exporta-lo para a plataforma Android.

Saída e pós condições: O aplicativo android (.app) pronto para ser publicado na Play Store.

[RF1.02] Gerar uma versão para dispositivos móveis iOS.

A aplicação deve funcionar em dispositivos móveis com sistema iOS.

Atores: Usuário

Prioridade: Essencial

Entradas e pré-condições: Usuário deve criar um novo projeto, configura-lo e exporta-lo para a plataforma iOS.

Saída e pós condições: O aplicativo iOS (.app) pronto para ser publicado na Apple Store.

[RF1.03] Gerar uma chave para assinar seu app

A aplicação deve gerar a chave de assinatura do seu android para que seja possível a publicação na play store

Atores: Usuário

Prioridade: Importante

Entradas e pré-condições: O usuário na hora de exportar deve entrar com as informações necessárias para criação da chave

Saída e pós condições: Uma chave para assinatura do aplicativo

[RF1.04] Ser capaz de ler um arquivo de configuração gerado pela ferramenta de criação de jogos CGT.

A aplicação deve ser capaz de ler um arquivo de configuração gerado pela ferramenta de construção de jogos CGT. E assim o jogo (imagem de fundo, sons, animação)

Atores: Usuário

Prioridade: Essencial

Entradas e pré-condições: um arquivo .cgt

Saída e pós condições: A leitura do .cgt com suas devidas configurações

2. Configurações Gerais

Todos esses requisitos necessitam de que a ferramenta esteja instalada previamente.

[RF2.01] Criar Screen.

Criar uma tela do jogo e configurar background. Pode ser a criação de uma tela de menu ou configurações.

Atores: Usuário

Prioridade: Essencial

[RF2.02] Adicionar botões para uma screen.

Adicionar botões para uma screen e configurar textures, posicionamento e ação.

Atores: Usuário

Prioridade: Essencial

[RF2.03] Criar Mundo.

Criação do jogo e configurar background.

Atores: Usuário

Prioridade: Essencial

[RF2.04] Configurar Popup de Pausa, Derrota e Vitória.

Configurar um Popup para cada um dos momentos. Isso inclui a configuração de posicionamento do Popup, texturas do Popup e botões que ele irá possuir.

Atores: Usuário

Prioridade: Essencial

Entradas e precondições: imagem ser do tipo .png

[RF2.05] Adicionar critérios de vitória.

Selecionar critérios de vitórias para o seu jogo e configurá-los.

Atores: Usuário

Prioridade: Essencial

[RF2.06] Adicionar critérios de derrota.

Selecionar critérios de derrotas para o seu jogo e configurá-los.

Atores: Usuário

Prioridade: Essencial

[RF2.07] Adicionar spritesheet.

Adicionar um spritesheet e configurá-lo informando linha e coluna.

Atores: Usuário

Prioridade: Essencial

[RF2.08] Adicionar música ao mundo.

Adicionar a música ao jogo que será executada como tema, música de derrota e vitória.

Atores: Usuário

Prioridade: Essencial

Entradas e precondições: o áudio precisa ser no formato .mp3

[RF2.09] Configuração dos atributos de objetos do jogo: Personagem principal.

Deverá ser possível a configuração dos atributos referentes ao personagem principal do jogo. Entre elas: vida, velocidade, posicionamento, colisão, invencibilidade, entre outras.

Atores: Usuário

Prioridade: Essencial

Entradas e precondições: Ter um novo projeto criado.

Saída e pós condições: O personagem principal do jogo configurado

[RF2.10] Adicionar action ao personagem principal.

Adicionar uma ação ao personagem principal configurando qual vai ser o tipo de entrada e a que movimento essa entrada está associada.

Atores: Usuário

Prioridade: Essencial

Saída e pós condições: Uma action foi adicionada ao personagem principal do jogo.

[RF2.11] Configuração dos atributos de objetos do jogo: Inimigos.

Deverá ser possível a configuração dos atributos referentes ao inimigos do jogo. Entre elas: vida, velocidade, posicionamento, colisão, invencibilidade, entre outras.

Atores: Usuário

Prioridade: Essencial

Entradas e precondições: Ter um novo projeto criado.

Saída e pós condições: Os inimigos do jogo configurado

[RF2.12] Adicionar comportamento sino ao inimigo.

Adiciona o comportamento ao inimigo selecionado e configura-se informando o tipo de movimentação, mínimo e máximo.

Atores: Usuário

Prioridade: Essencial

Saída e pós condições: Um comportamento adicionado ao inimigo

[RF2.13] Adicionar comportamento fade ao inimigo.

Adiciona o comportamento ao inimigo selecionado e configura-se informando o tempo em segundos.

Atores: Usuário

Prioridade: Essencial

Saída e pós condições: Um comportamento adicionado ao inimigo

[RF2.14] Adicionar comportamento em 8 e 4 direções.

Adiciona o comportamento ao inimigo selecionado e configura-se informando o direction mode, x máximo e mínimo e y máximo e mínimo.

Atores: Usuário

Prioridade: Essencial

Saída e pós condições: Um comportamento adicionado ao inimigo

[RF2.15] Adicionar comportamento em 2 direções

Adiciona o comportamento ao inimigo selecionado e configura-se informando o direction mode, posição inicial e posição final.

Atores: Usuário

Prioridade: Essencial

Saída e pós condições: Um comportamento adicionado ao inimigo

[RF2.16] Adicionar comportamento de onda ao inimigo.

Adiciona o comportamento ao inimigo selecionado e configura-se informando a amplitude do movimento, frequência, fase e o máximo ou mínimo valor para x dependendo da onde ou termina o movimento.

Atores: Usuário

Prioridade: Essencial

Saída e pós condições: Um comportamento adicionado ao inimigo

[RF2.17] Configuração dos atributos de objetos do jogo: obstáculos.

Deverá ser possível a configuração dos atributos referentes aos inimigos do jogo. Entre elas: posicionamento, colisão, bloqueante, destrutível, comportamentos, entre outras.

Atores: Usuário

Prioridade: Essencial

Entradas e pré condições: Ter um novo projeto criado.

Saída e pós condições: Os obstáculos do jogo configurados

[RF2.18] Configuração dos atributos de objetos do jogo: bonus.

Deverá ser possível a configuração dos atributos referentes ao bonus do jogo. Bonus é tudo aquilo que pode beneficiar o personagem principal, ou para mudar a dinâmica do jogo. Os atributos podem ser: posicionamento, colisão, bloqueante, destrutível, comportamentos, entre outras.

Atores: Usuário

Prioridade: Essencial

Entradas e precondições: Ter um novo projeto criado.

Saída e pós condições: Os bonus do jogo configurado.

[RF2.19] Adicionar animação.

Adicionar animação e configurá-la selecionando spritesheet, estados que vão estar associados a animação, início e fim da animação e política de animação.

Atores: Usuário

Prioridade: Essencial

Entradas e precondições: Ter um spritesheet já cadastrado

Saída e pós condições: animação configurada no objeto selecionado

[RF2.20] Configuração dos atributos de objetos do jogo: projéteis.

Deverá ser possível a configuração dos atributos referentes aos projéteis do jogo. Projétil é todo objeto que é anexado à algum objeto do jogo, podendo ou não, causar algum dano. Os atributos podem ser: posicionamento, colisão, dano, munição, intervalo de lançamento, comportamentos, entre outras.

Atores: Usuário

Prioridade: Essencial

Entradas e pré-condições: Ter um novo projeto criado.

Saída e pós condições: Os projéteis do jogo configurado.

[RF2.21] Configurar posicionamento do projétil .

Configurar posicionamento do projétil em relação ao ator principal associando ao estado do ator.

Atores: Usuário

Prioridade: Essencial

Saída e pós condições: Posicionamento configurado do projétil

[RF2.22] Adicionar HUD de munição.

Configurar a exibição da munição na tela configurando posicionamento na tela, ícone e de qual projétil será mostrado a munição

Atores: Usuário

Prioridade: Essencial

Entradas e pré-condições: Ter um novo projeto criado.

Saída e pós condições: O HUD configurado exibido na tela.

[RF2.23] Adicionar HUD de inimigos.

Configurar a exibição de uma barra na tela sobre o número de inimigos destruíveis configurando posicionamento na tela, textura de fundo e textura da barra

Atores: Usuário

Prioridade: Essencial

Entradas e pré-condições: Ter um novo projeto criado.

Saída e pós condições: O HUD configurado exibido na tela

[RF2.24] Adicionar HUD de lifebar de um objeto.

Configurar a exibição do life de um objeto na tela configurando posicionamento na tela, textura de fundo, textura da barra e de qual vai ser exibido o life.

Atores: Usuário

Prioridade: Essencial

Entradas e pré-condições: Ter um novo projeto criado.

Saída e pós condições: O HUD configurado exibido na tela

[RF2.25] Adicionar HUD de tempo

Configurar a exibição do tempo na tela configurando posicionamento na tela.

Atores: Usuário

Prioridade: Essencial

Entradas e pré-condições: Ter um novo projeto criado.

Saída e pós condições: O HUD configurado exibido na tela.

Requisitos não Funcionais

O documento de requisitos não funcionais mostra as restrições aos serviços ou funções oferecidas pelo sistema, como: restrições no processo de desenvolvimento e restrições impostas pelas normas. Esses requisitos se aplicam ao sistema como todo. Também podemos observar nessa parte do documento características relacionadas ao *Ceará Game Tools* como usabilidade, confiabilidade, tempo de resposta entre outras.

[NF01] Interface amigável

Prioridade: Desejável

O *Ceará Game Tools* irá oferecer interfaces gráficas com usuário bem desenvolvidas, visando a simplicidade e facilidade de manuseio para que ele encontre as funcionalidades desejadas. Incluindo, também, um fluxo de telas facilmente lógico e fluído.

[NF02] Rotina de testes

Prioridade: Importante

Serão realizados diversos testes de sistema para garantir a confiabilidade do *Ceará Game Tools*. Uma série de testes trará maior certeza que às informações inseridas no sistema terão tratamento e respostas esperadas. Tanto teste de comportamento como teste unitários e de integração.

[NF03] Aplicação rápida

Prioridade: Importante

A aplicação deverá ser leve para que possa ser instalada em qualquer aparelho principalmente em sistemas operacionais menos robustos. Deverá consumir poucos recursos de processamento e memória assim como armazenamento em disco, para que o requisito seja atendido.

[NF04] Sistema escalável

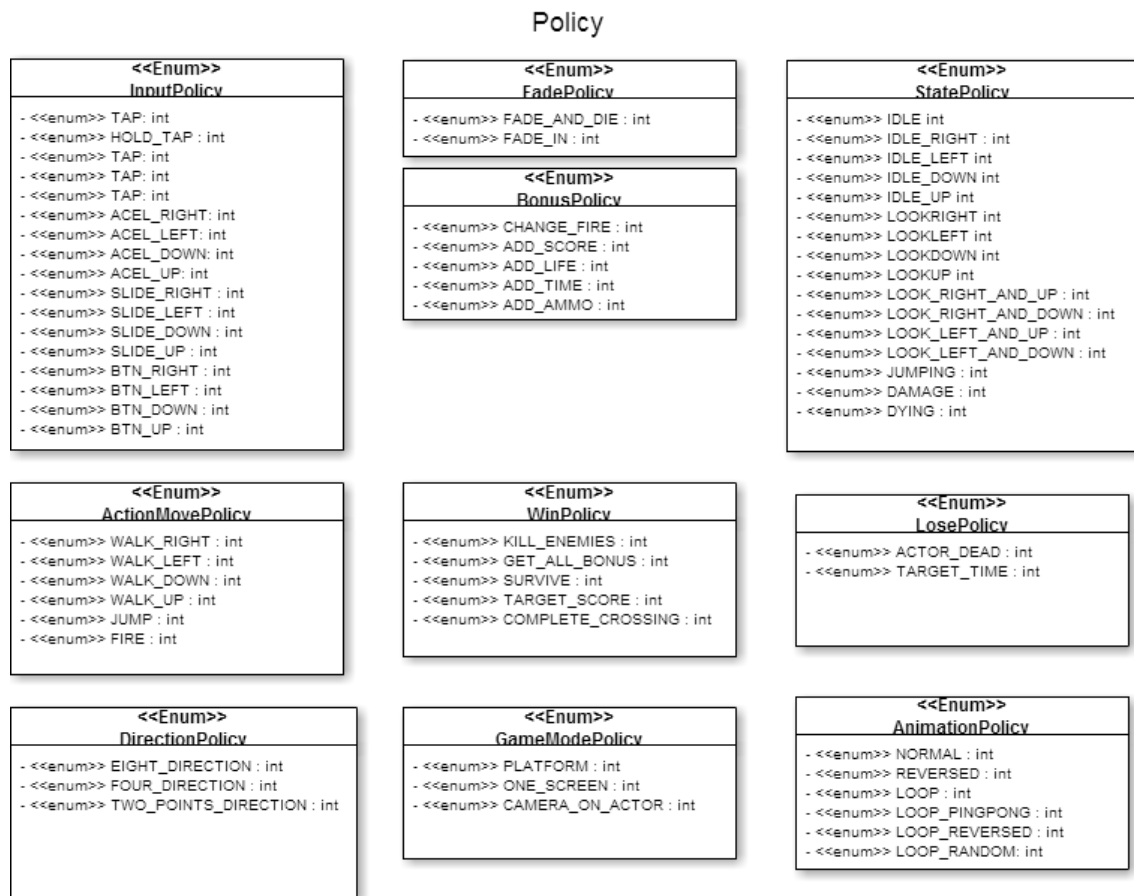
Prioridade: Importante

Com base na documentação do sistema, definições de funcionalidades e legibilidade o sistema foi dividido com o objetivo de compreensão da estrutura do software.

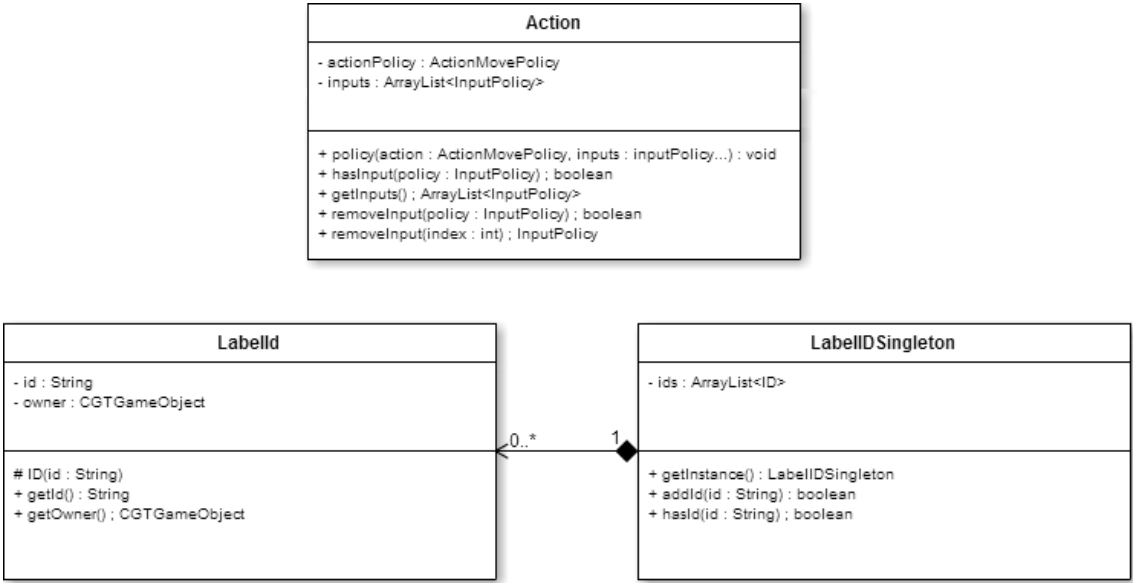
DIAGRAMA DE CLASSE

O diagrama de classes representa a estrutura do sistema, recorrendo ao conceito de classe e suas relações. O modelo de classes resulta de um processo de abstração onde são identificados os objetos relevantes do sistema em estudo. Para facilitar a legibilidade das classes e para que o sistema possa ser desenvolvido de modo organizado, a apresentação dos diagramas será feita por pacotes.

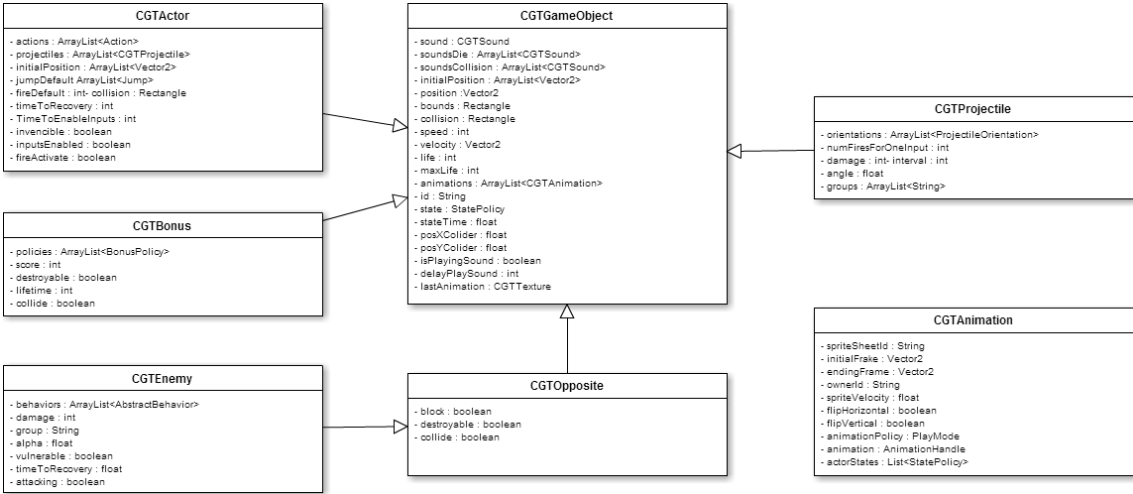
Pacote *policy*



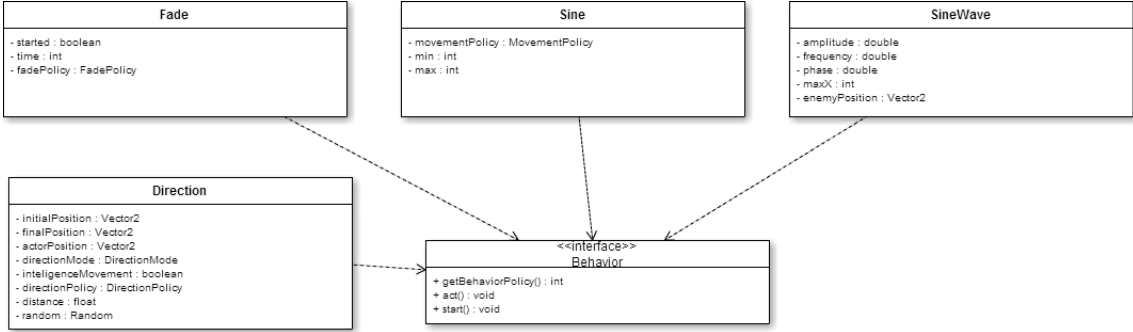
Pacote unit



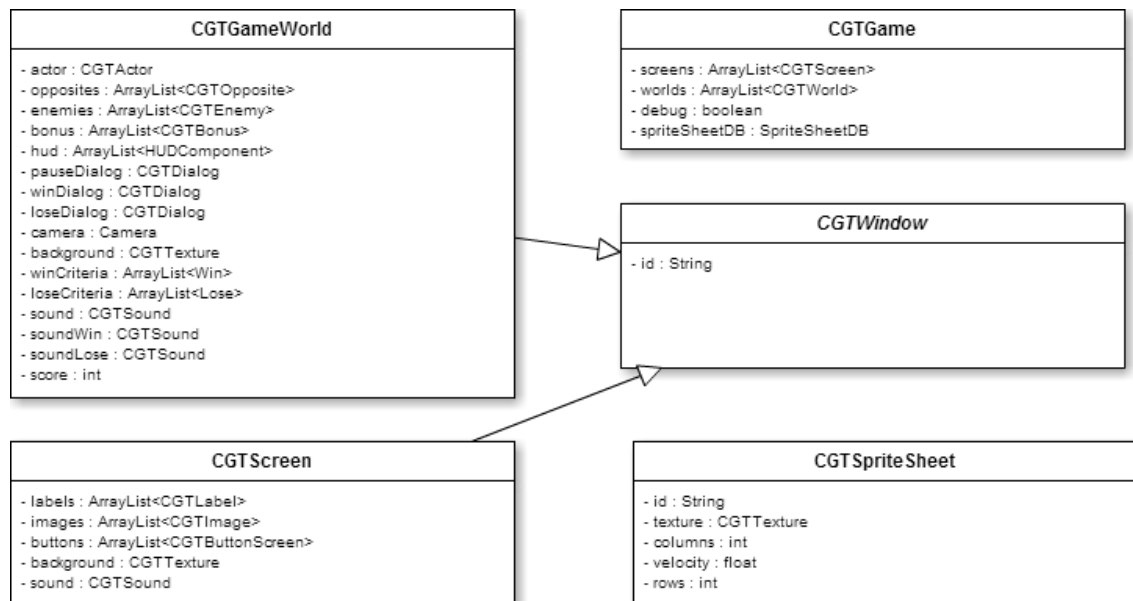
Pacote core



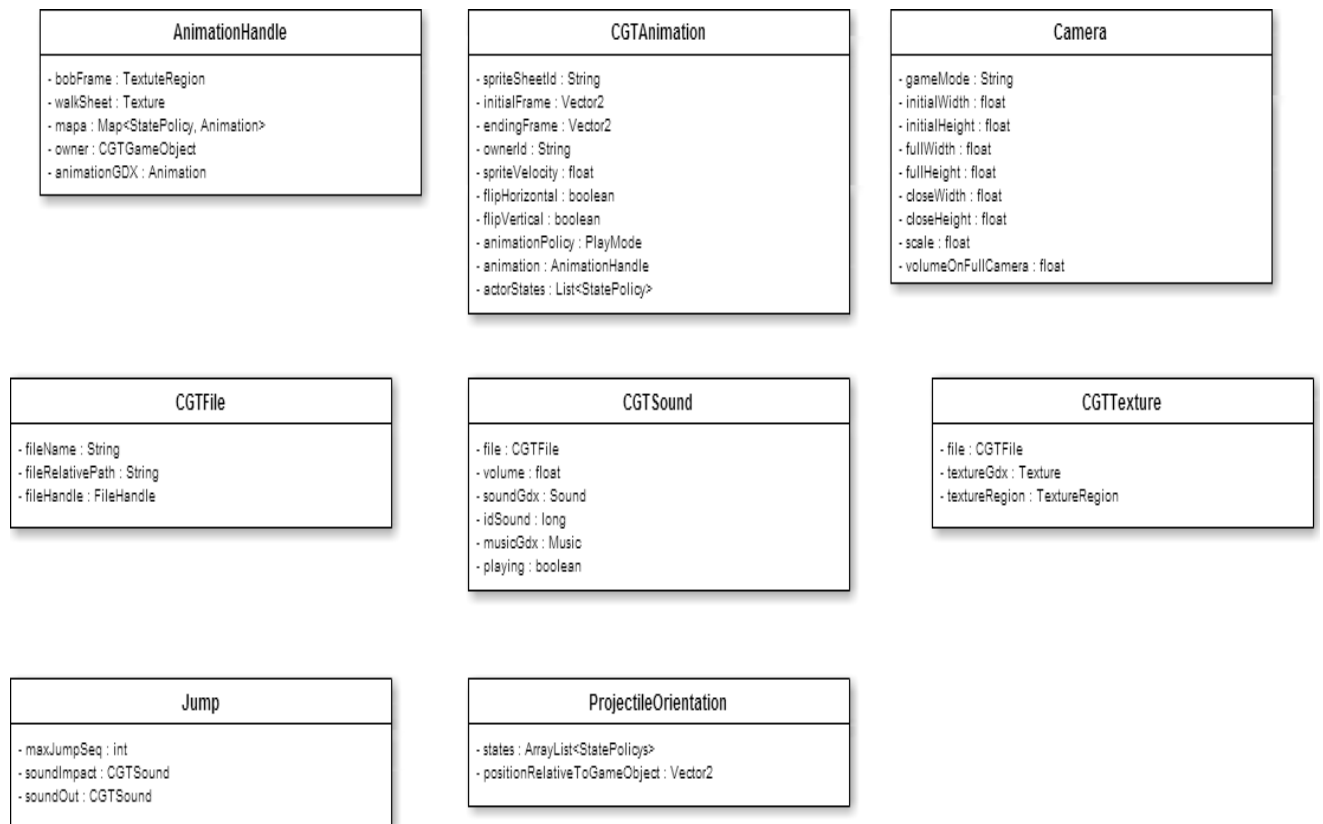
Pacote behaviors



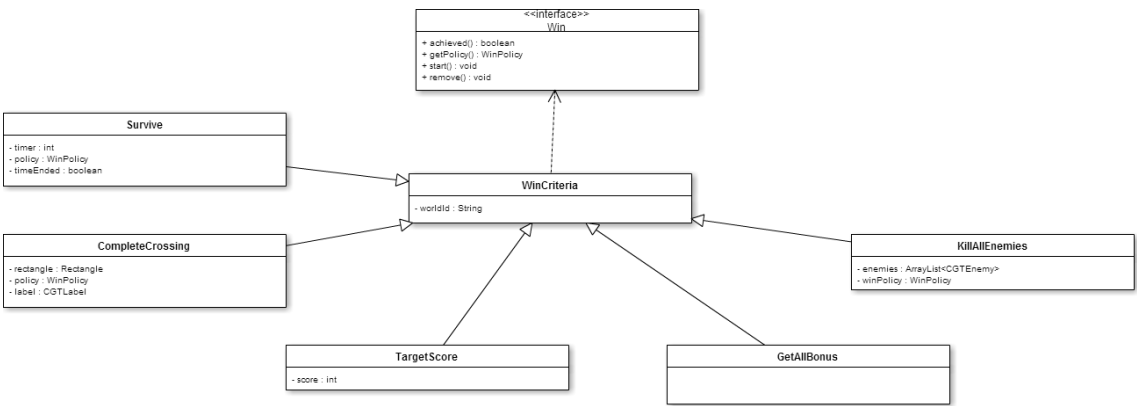
Pacote *cgt*



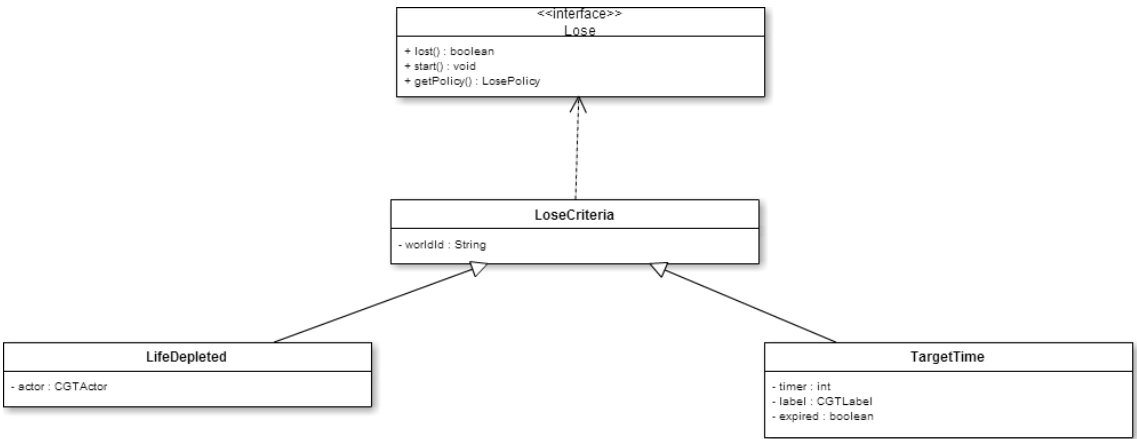
Pacote Util



Pacote Win



Pacote Lose



Anexo 2

Sumário

1.	Configuração de um jogo com a ferramenta.....	39
1.1.	Criar <i>Screen</i>	39
1.2.	Criar Mundo	44
1.2.1.	Configurar <i>Pop-ups</i>	45
1.2.2.	Critérios de Vitória e Derrota.	46
1.2.3.	Ator Principal.....	46
1.2.4.	Configurar Opositores	50
1.2.5.	Configurar Inimigo.....	51
1.2.6.	Bônus.....	53
1.2.7.	Configurar Projétil	53
1.2.8.	Configurar <i>HUD</i>	54
2.	Tela por tela da ferramenta	57
2.1.	Configurar <i>Screen</i>	57
2.2.	Configurar Mundo.....	59
2.2.1.	Configurações Gerais	59
2.2.2.	Configurar <i>Pop-up</i>	61
2.2.3.	Configurar critérios de Vitória	62
2.2.4.	Critérios de Derrota	63
2.2.5.	Configurar <i>Spritesheet</i>	64
2.2.6.	Configurar Animação(Ator,Inimigo,Opositor,Bonus,Projétil)	65
2.2.7.	Configuração Comum dos Objetos(Ator,Inimigo,Opositor,Bonus,Projétil)	65
2.3.	Configurações Específicas do Ator	66
2.3.1.	Configuração das actions do ator.....	67
2.4.	Configurações Específicas dos Inimigos	67
2.4.1.	Configuração Comportamento Sino	68
2.4.2.	Configuração Comportamento <i>Fade</i>	68
2.4.3.	Configuração Comportamento Onda	69
2.5.	Configurações Específicas dos Opositores.....	69
2.6.	Configurações Específicas dos Bônus.....	70
2.7.	Configurar Projétil	71
2.7.1.	Configuração do Posicionamento	71
2.8.	Configurar <i>HUDS</i>	71

2.8.1.	Configuração do <i>Display</i> de Munição.....	71
2.8.2.	Configuração do <i>LifeBar</i> Inimigos.....	72
2.8.3.	Configuração <i>LifeBar</i> Objeto	73
3.	Executar projeto	74
4.	Configurar <i>Game</i>	75
5.	Exportar projeto	76
6.	Gerar Chave de Assinatura.....	77

Através deste manual será possível encontrar uma descrição sucinta da ferramenta CGT(Ceará Game Tools), mostrando todas as suas funcionalidades e imagens que explicam as suas configurações. Dividido em duas partes, a primeira parte deste manual é referente a configuração de um jogo usando a ferramenta, já a segunda parte se refere a explicação de cada funcionalidade que o CGT possui. Assim, o usuário possuirá um exemplo prático e a parte teórica caso já saiba o que deseja.

1. Configuração de um jogo com a ferramenta

Com o intuito de mostrar as funcionalidades que a ferramenta possui, será demonstrada a criação de um jogo com a ferramenta CGT.

1.1. Criar *Screen*

Uma *screen* representa uma tela do jogo inicial, podendo ser encontrada como uma tela de configurações, uma tela de créditos, entre outros exemplos. Para criar uma screen basta clicar em “Criar *Screen*” e fornecer um nome.

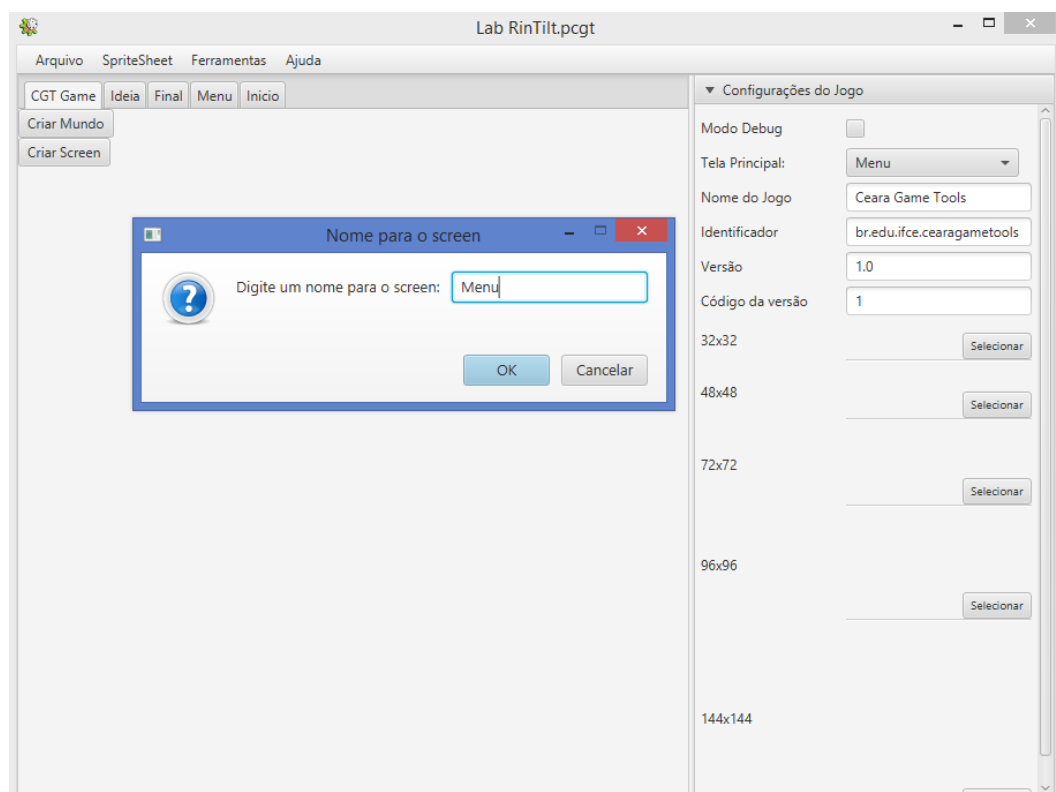


Figura 01 - Criar *Screen*.

Na tela de *screen* o usuário pode configurar três objetos: plano de fundo, áudio e botões. No exemplo a seguir foi selecionada uma imagem para o plano de fundo da *screen* e adicionado quatro botões a ela.

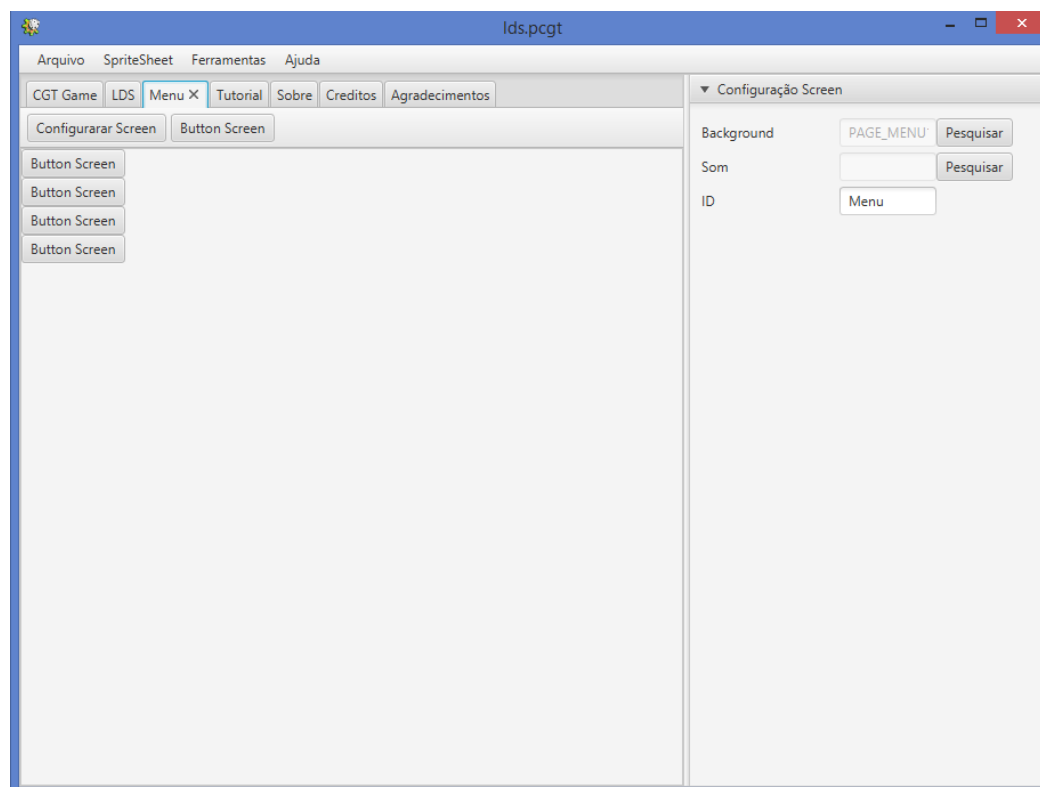


Figura 02 - Configurar Screen.

Agora serão configurados os botões que foram adicionados a *screen*. Como o procedimento é o mesmo para todos os botões, será demonstrada apenas a configuração de um. Nele serão configurados ação, imagem, posicionamento e dimensão do botão.

A ação do botão, representado pelo “ir para”, será selecionada outra *screen* previamente criada ou um Mundo, que será explicado ao longo do texto. Quanto a imagem do botão, será selecionada uma para quando se encontrar no estado de normal e outra para quando for pressionado, causando a ideia de efeito ao clicar sobre o botão. O seu posicionamento na tela e dimensão são configurados com base na *screen* e os valores devem variar de 0 a 1, representando a porcentagem da tela. No exemplo da figura 3, o botão ficará posicionado a 30% da tela no eixo X e 4% da tela no eixo Y, tendo 40% da largura da *screen* e 13% da altura da *screen*.

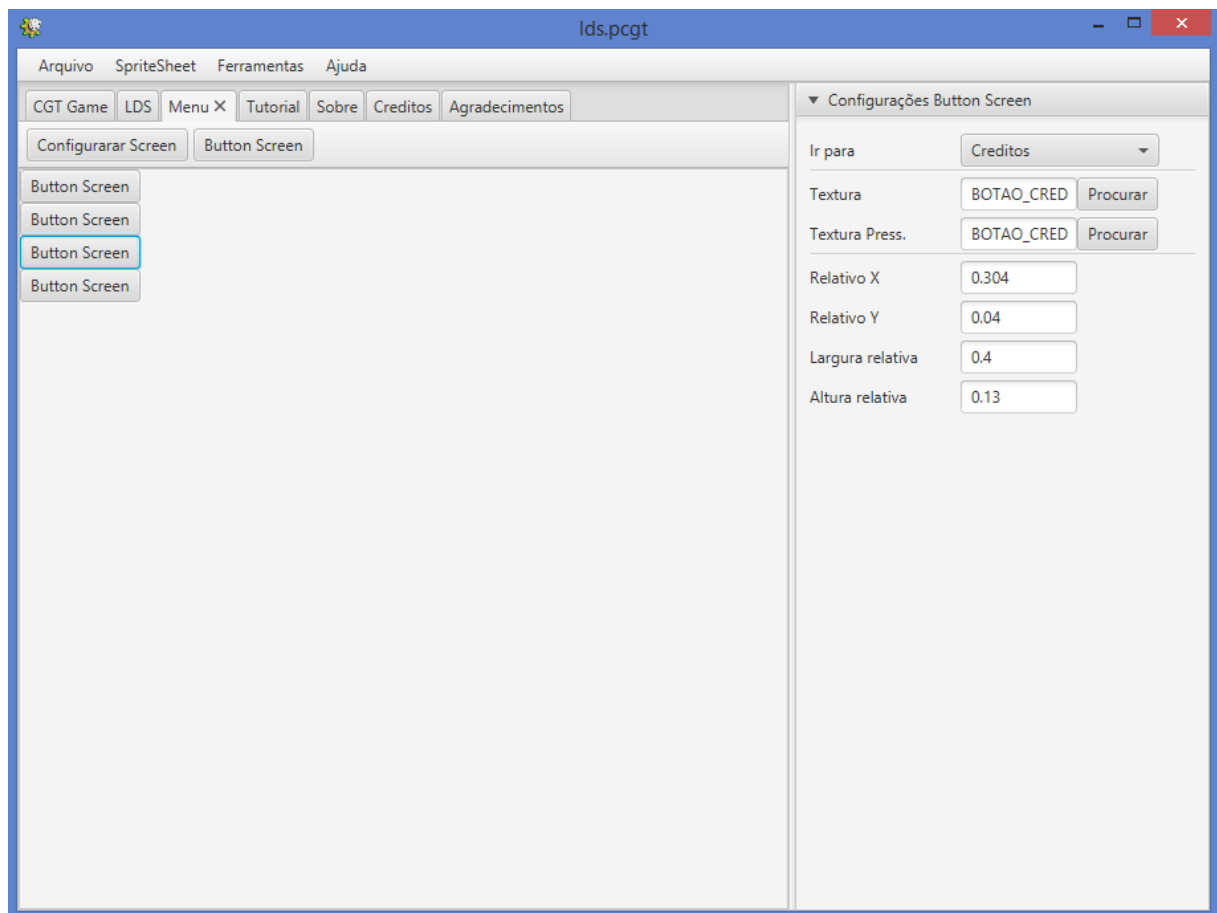


Figura 03 - Configurar Botão na Screen.

O mesmo procedimento irá ocorrer para os outros botões, assim como para as outras *screens* criadas: “Tutorial”, “Sobre”, “Créditos” e “Agradecimentos”. Segue abaixo as imagens das outras *screens* criadas.



Figura 04 - Menu Inicial do Jogo.

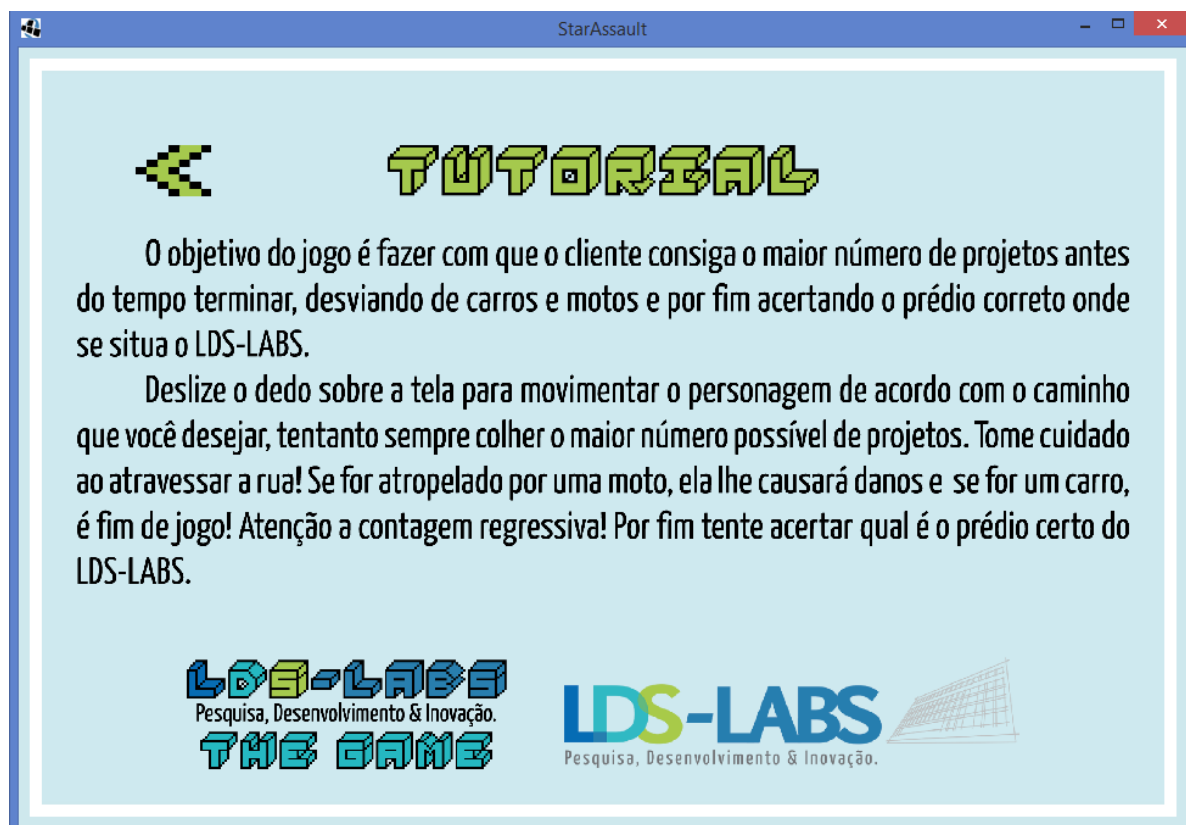


Figura 05 - Tela de Tutorial.

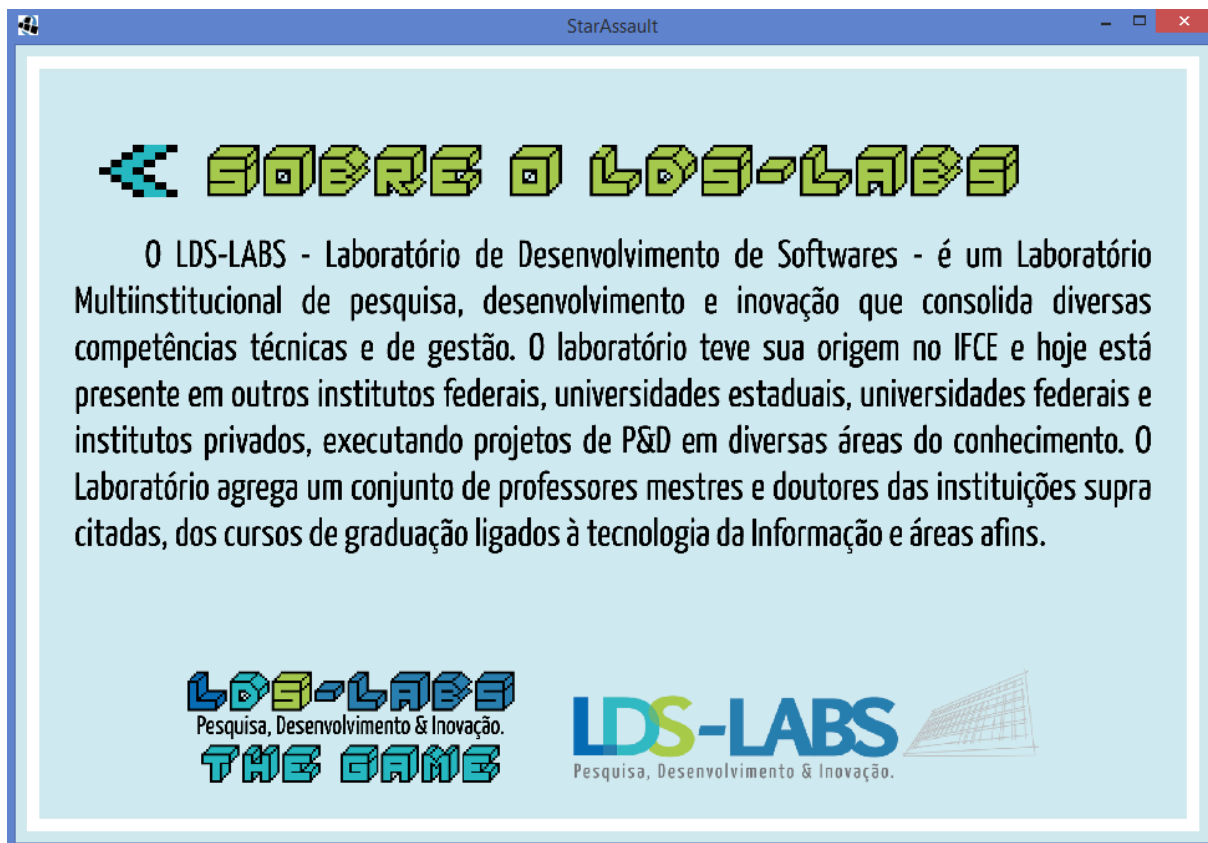


Figura 06 - Tela de Sobre.

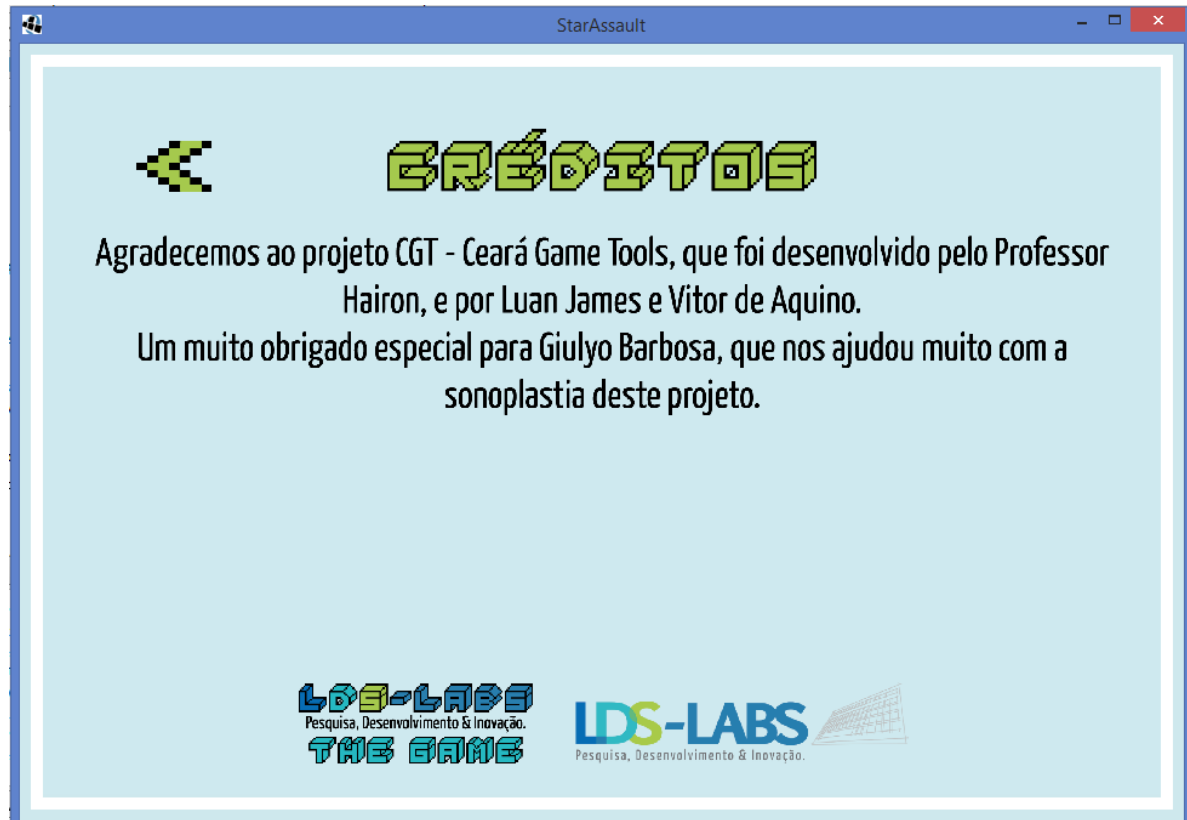


Figura 07 - Tela de Créditos.

1.2. Criar Mundo

O primeiro botão da screen inicial que foi criada leva a um Mundo. Um mundo pode ser o seu jogo ou uma fase dele e é constituído de vários objetos que serão configurados mais adiante. Para criar um mundo basta clicar em “Criar Mundo” e fornecer um nome.

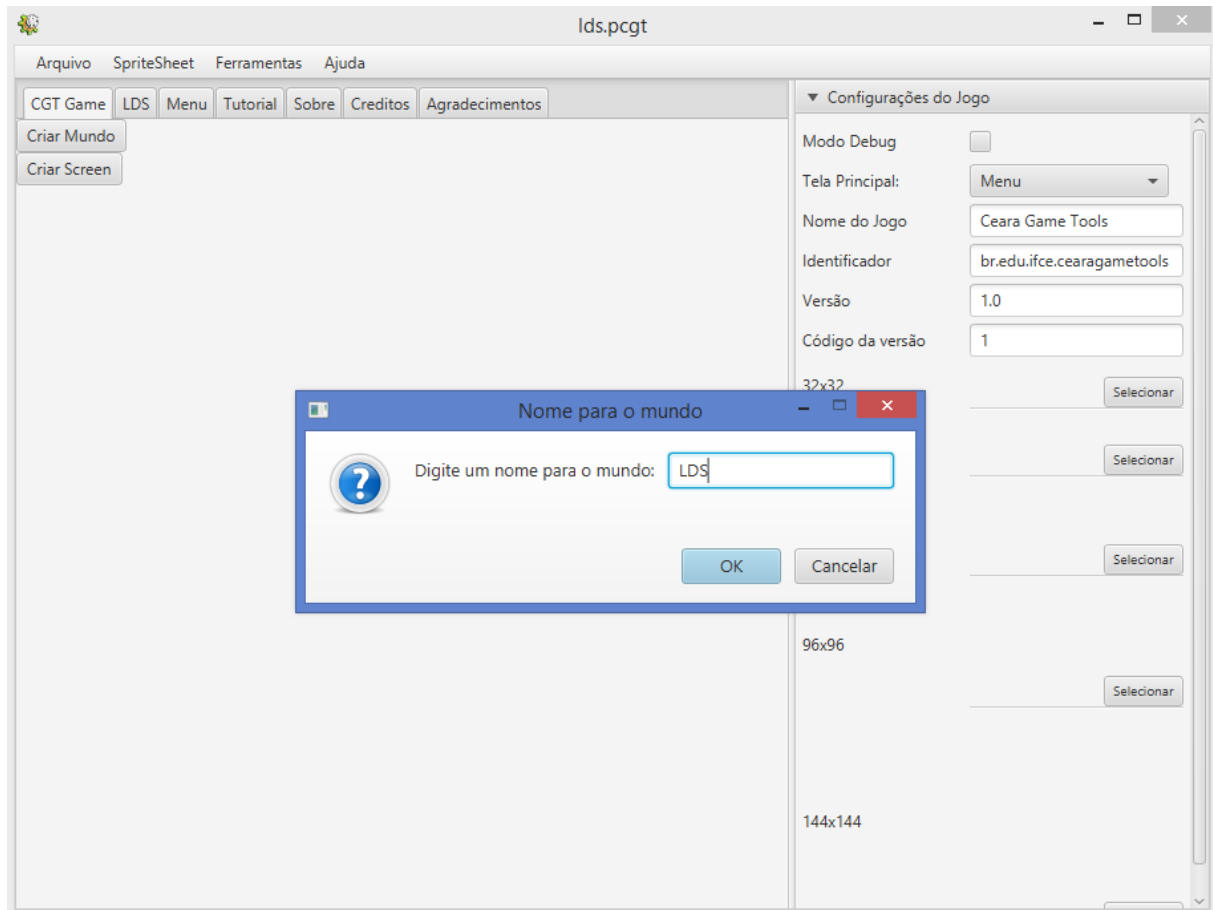


Figura 08 - Criar Mundo.

Como mostrado na imagem a seguir, o mundo possui algumas características gerais. Ele deve possuir um plano de fundo, *pop-up* de pausa (tela que será exibida quando o jogo for pausado), *pop-up* de derrota (tela que será mostrada quando o jogo tiver atingido algum um critério de derrota), *pop-up* de vitória (tela que será visualizada quando for atingido algum critério de vitória) e áudio, caso queira que seu jogo possua um.

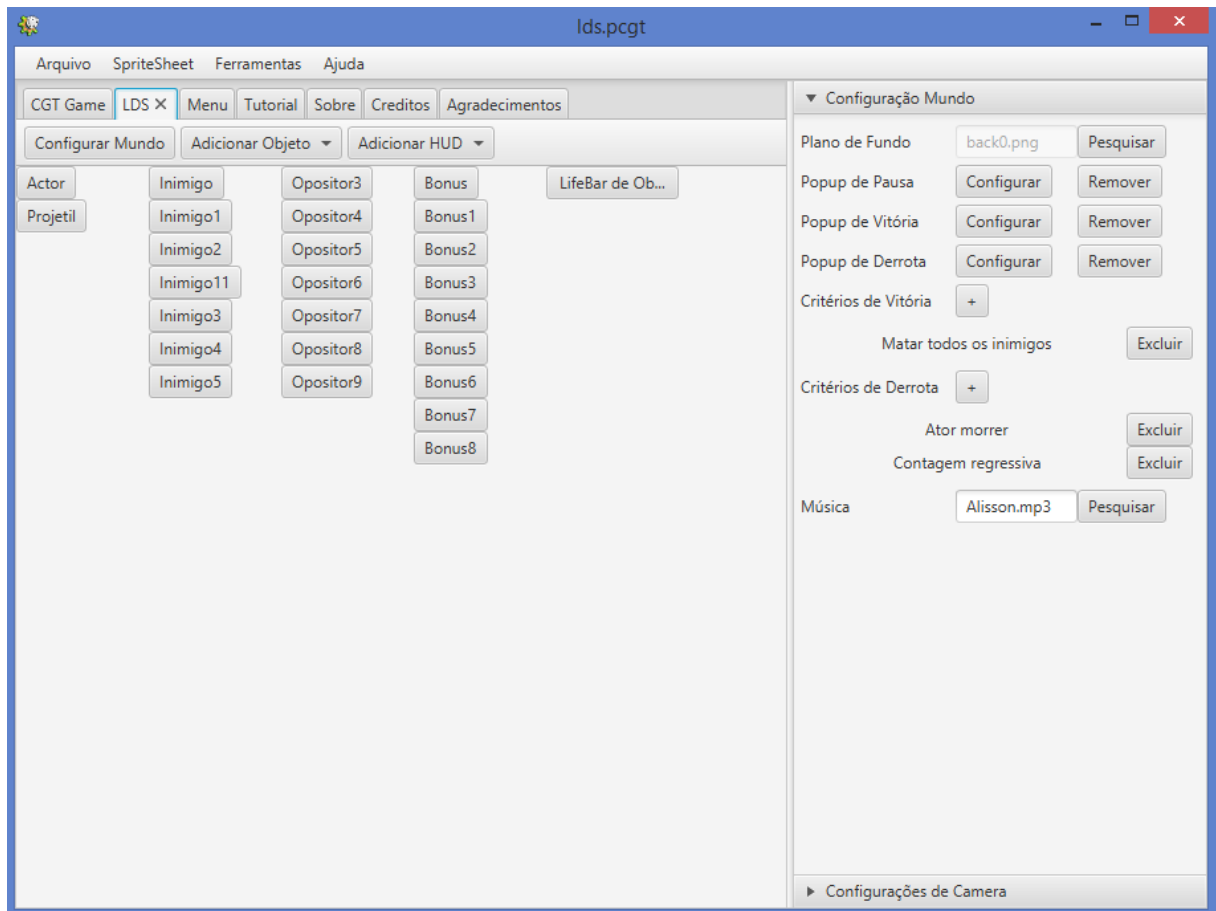


Figura 09 - Configurações Gerais do Mundo.

1.2.1. Configurar *Pop-ups*

A configuração dos *Pop-ups* são todas iguais, então será demonstrada apenas a configuração um deles. Para configurar o *Pop-up* desejado, basta clicar em “configurar”, será necessário ajustar posicionamento, dimensão, imagens e botões. O posicionamento, as dimensões e os botões, possuem os mesmos modos de configuração explicados no tópico 1.1.

O *Pop-up* deve possuir três imagens bases, que precisam ser do tipo .png, são elas o fundo, a borda horizontal e a borda do canto inferior. A imagem de fundo será o plano de fundo do *pop-up*. As imagens de borda horizontal e canto inferior direito são imagens usadas para se definir a borda de todo o *pop-up*.

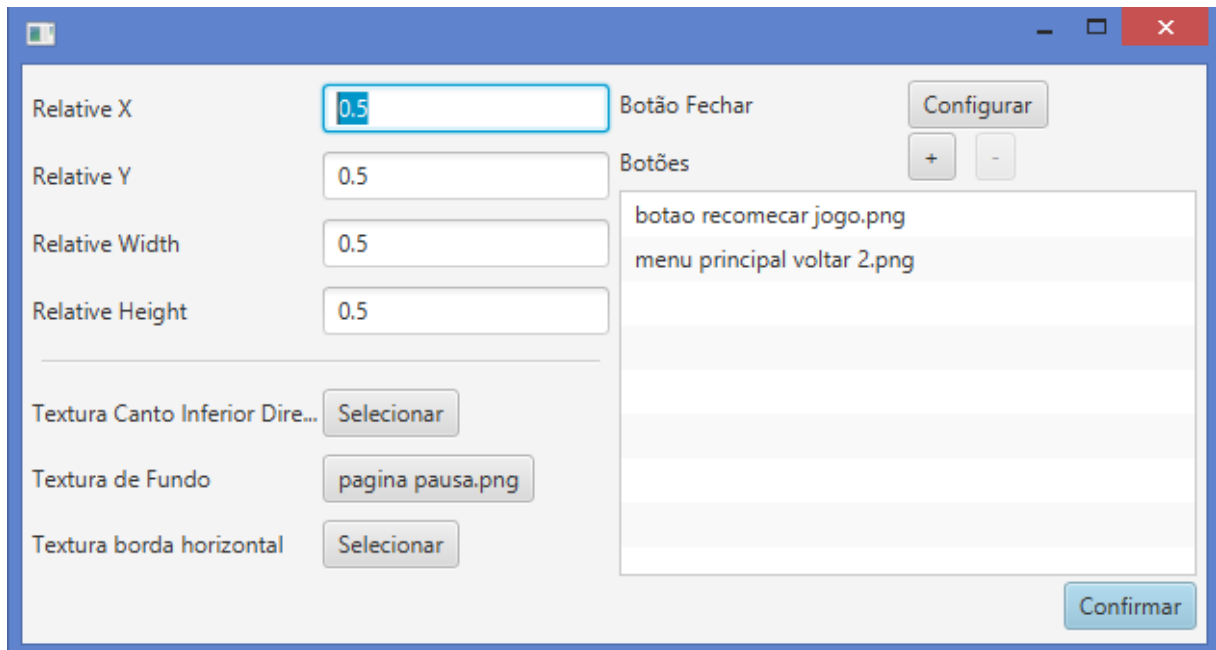


Figura 10 - Configuração *Pop-up*.

1.2.2. Critérios de Vitória e Derrota.

Para adicionar critérios de derrota e vitória basta clicar no “+” e escolher os critérios desejados. No exemplo demonstrado o critério de derrota selecionado foi “O ator morrer” e “Contagem Regressiva”, como pode ser visto na figura 09. O critério do “ator morrer” é atingido quando o ator principal possuir zero ou menos de vida, já o de “contagem regressiva” é atingido quando o tempo configurado do critério acabar. O critério de vitória é “Matar todos os inimigos”, ou seja, quando todos os inimigos destrutíveis adicionados ao mundo possuírem vida menor ou igual a zero, o critério será atingido.

1.2.3. Ator Principal.

Cada Mundo deve possuir um ator principal, que é aquela personagem que o usuário vai poder controlar. O ator principal possui características em comuns com os outros objetos que o Mundo também possui (Inimigo, Opositor, Bônus, Projétil).

A primeira característica a ser configurada vai ser a animação. A animação é feita através de uma imagem que possui várias outras imagens menores dentro dela, chamada de *spritesheet*. Primeiramente será adicionado um *spritesheet* clicando em “*spritesheet*” e depois em “adicionar”. Para configurar um *spritesheet* é preciso selecionar uma imagem, inserir um nome e informar o número de linhas e colunas que a imagem possui.



Figura 11 - Configurar *Spritesheet*.

Com o *spritesheet* adicionado, será configurada a animação. Para isto, deve-se ajustar estado do ator, seleccionar *spritesheet*, informar *frame* inicial e final, velocidade e seleccionar uma política de animação. O estado do ator serve para relacionar o estado escolhido com a animação. O *frame* inicial e o final são as imagens iniciais e finais da animação configurada, que são indexadas a partir do zero, ou seja, as colunas e linhas começam a ser contadas a partir do zero. A velocidade é relacionada à troca de imagens. A política de animação é a forma como animação irá ocorrer.

No caso da imagem a seguir, está sendo configurada a animação de andar para baixo. Como pode ser visto a animação de andar para baixo fica na linha 4. Como é indexado a partir do zero, o *frame* inicial seleccionado foi a imagem do meio da linha 4 e a última imagem da linha 4.

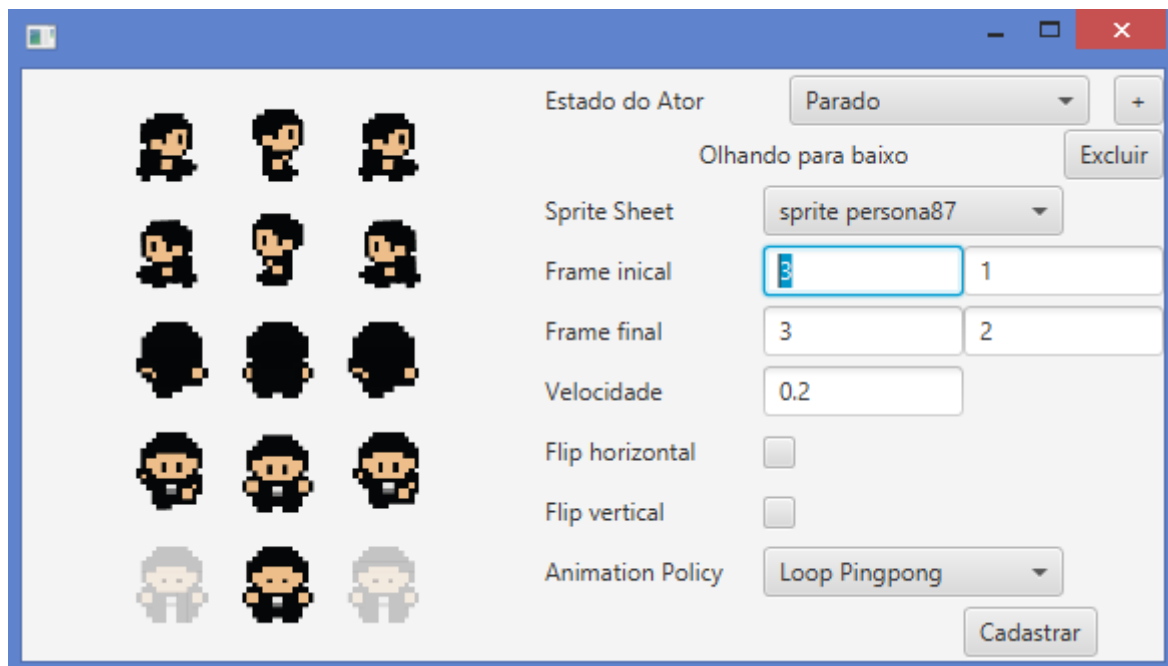


Figura 12 - Configurar Animação.

Agora será configurado o resto das características do ator. Como pode ser visto na figura 13, foi determinado que a vida máxima do ator principal será 50, assim como a vida inicial. Além disso, foram adicionadas duas posições iniciais, ou seja, será sorteado ao iniciar o jogo qual das posições o ator principal vai iniciar. Também foi determinado que o ator irá possuir 200x200 *pixels* e que a caixa de colisão também possuirá também 200x200 *pixels*. A velocidade do ator foi definida para 400, ou seja, ele anda 400 *pixels* por segundo.

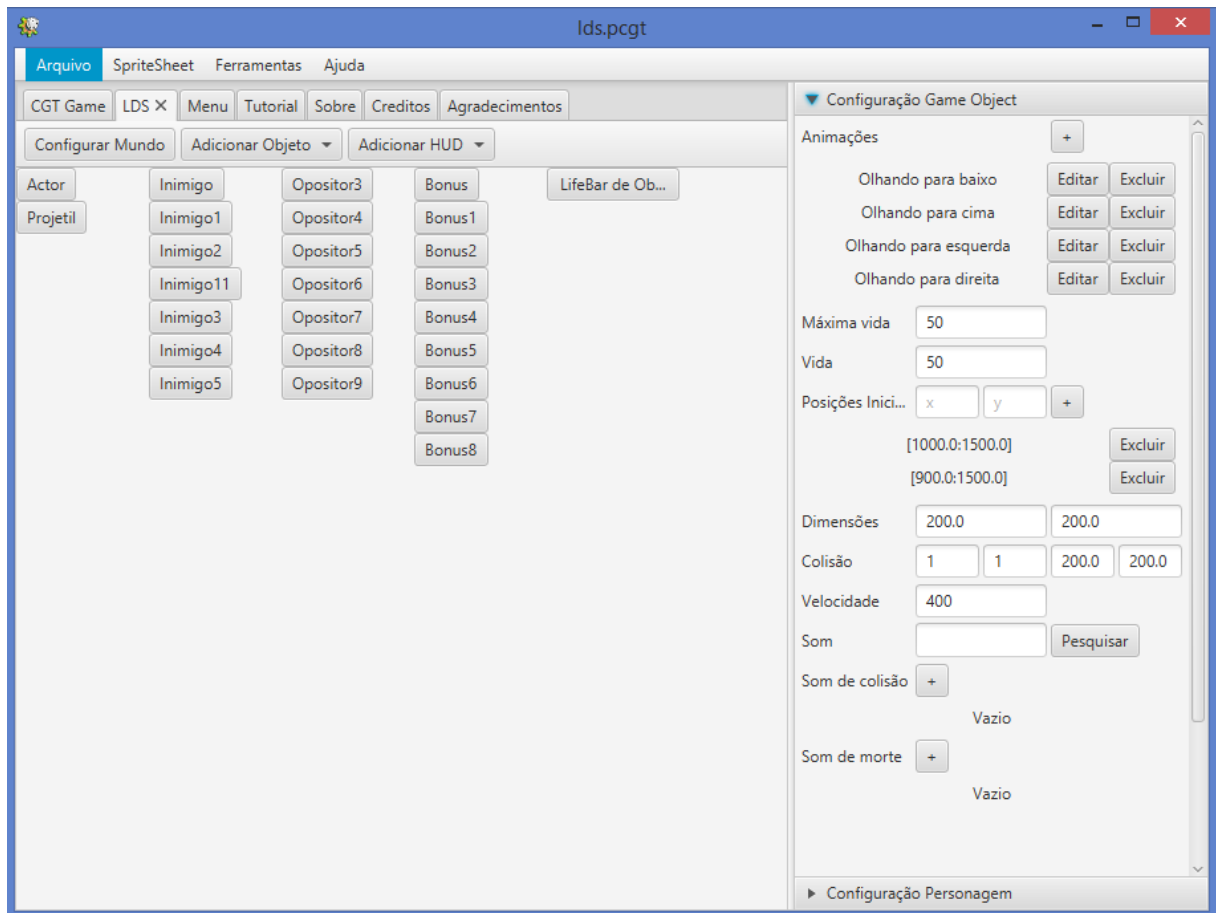


Figura 13 : Configurações Gerais Ator.

Nas configurações específicas do ator são configuradas as ações, que são entradas para controle do ator. No exemplo da figura 14, foi configurado que ao deslizar o dedo para cima na tela do celular o personagem irá andar para cima, as outras ações seguem a mesma lógica. Além disso, foi adicionado um projétil padrão ao ator que será explicado mais adiante. O tempo de recuperação é o tempo em segundos que o ator, ao colidir com um inimigo, irá demorar para poder se movimentar novamente.

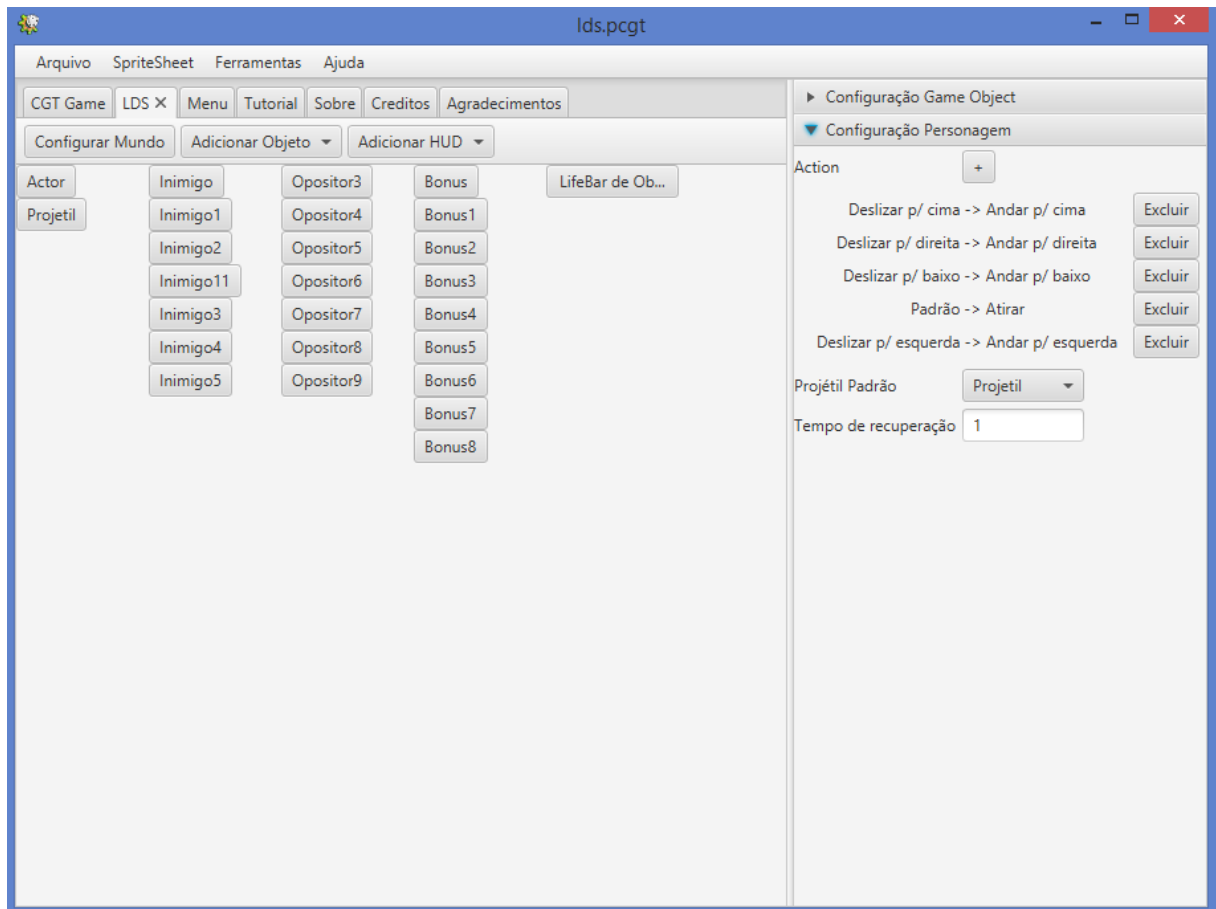


Figura 14 - Configurações Específicas do Ator.

1.2.4. Configurar Opositores

Opositores são objetos comuns do jogo, que podem ser casas, paredes, árvores e entre outros exemplos. A seguir, será demonstrada as suas configurações específicas, que são *block* e destruível. O *block* define se o ator principal pode se colidir com ele. Destruível define se o ator pode “matar” o inimigo com o projétil.

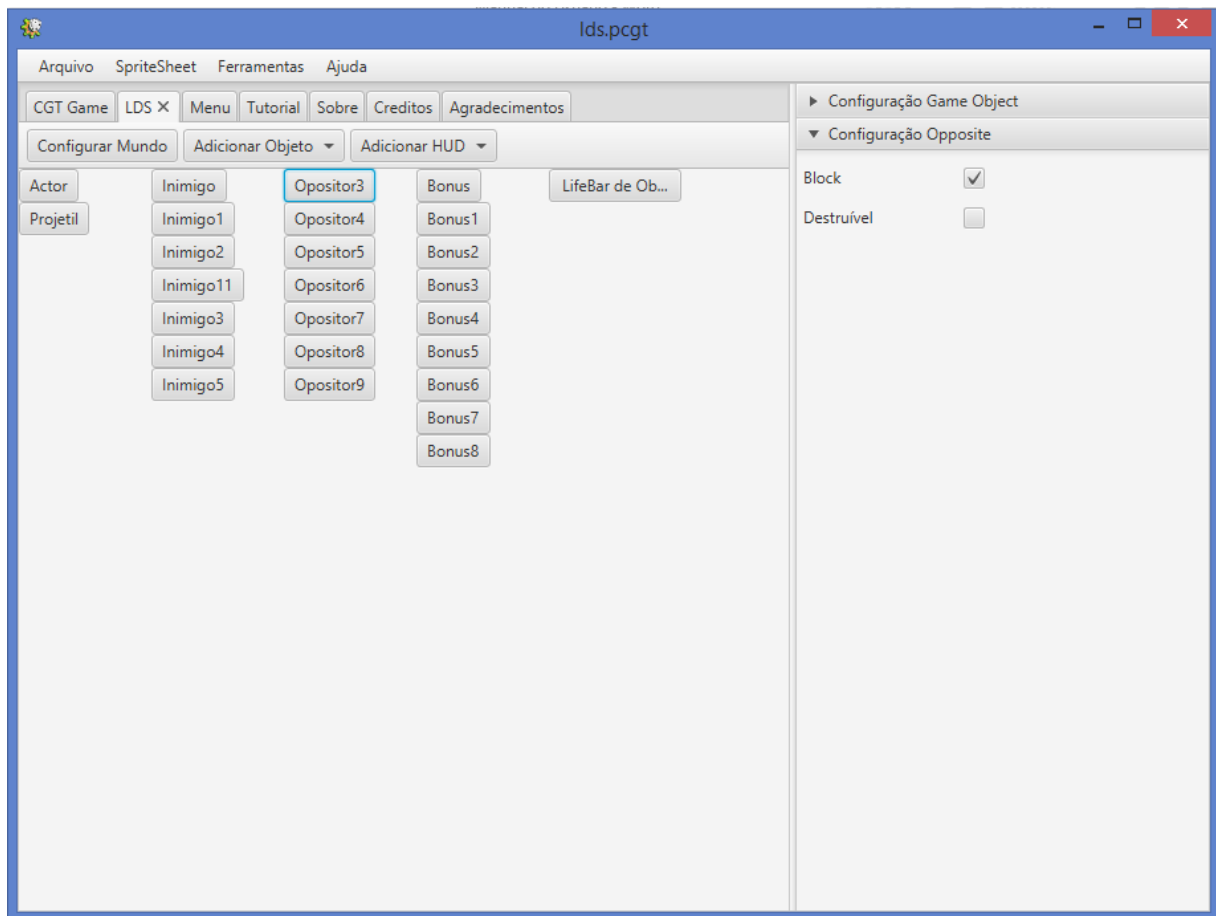


Figura 15 - Configurações Específicas do Opositor.

1.2.5. Configurar Inimigo

Inimigos no jogo são opositores, que podem se movimentar e causar dano ao ator principal. Suas características específicas são *block*, destruível, comportamento, valor do dano e grupo. *Block* e Destruível foram citados no tópico 1.2.4. Comportamento define a maneira como o inimigo irá se movimentar. Valor do dano define quanto o inimigo irá retirar de vida do ator principal ao colidir. O grupo garante que se o inimigo e o ator principal possuírem o mesmo grupo, eles poderão causar dano um no outro.

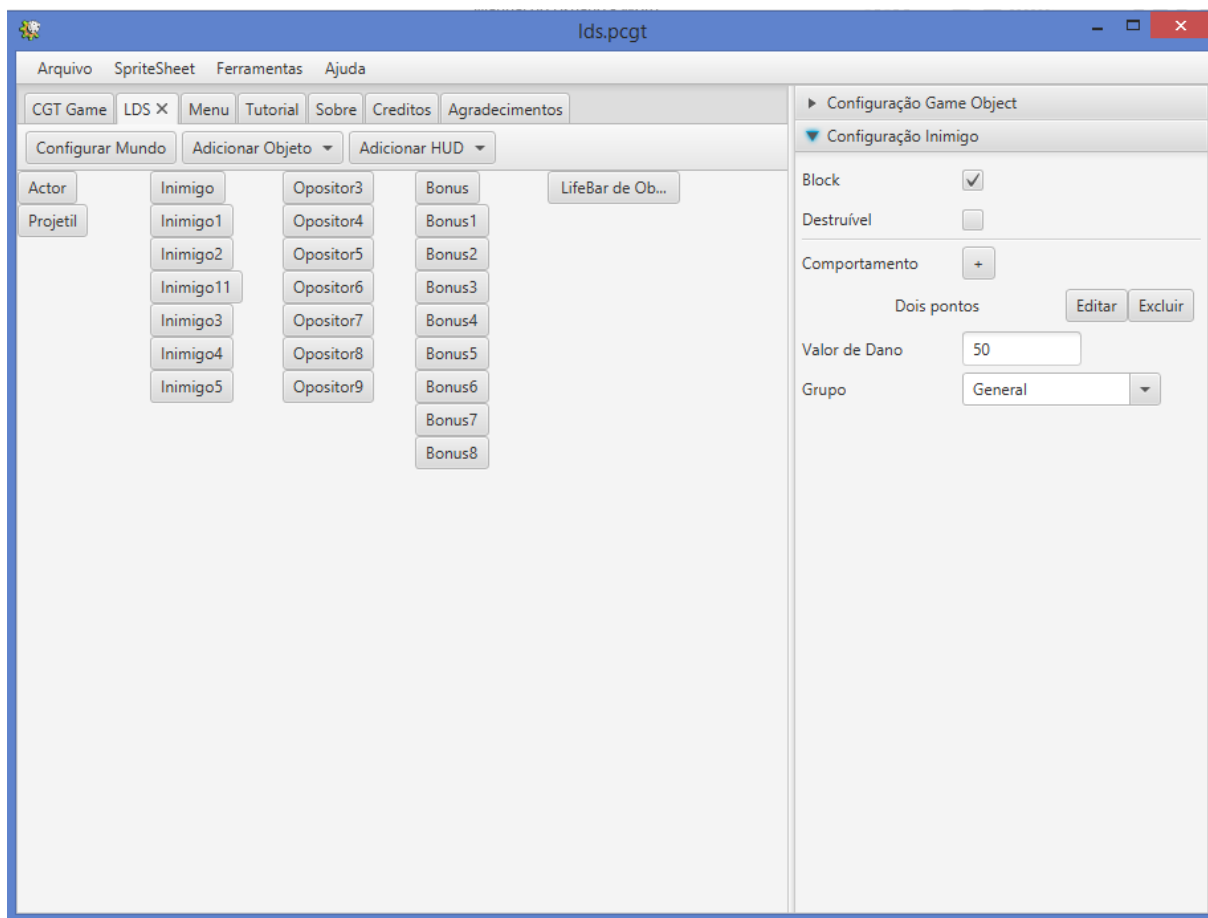


Figura 16 - Configurações Específicas do Inimigo.

A configuração de movimentação definida foi a “dois pontos”. Com essa movimentação o inimigo irá se movimentar entre a posição inicial e a final com a política de loop, ou seja, ele irá do ponto inicial para o final e depois retornará para o ponto inicial para começar a movimentação novamente.

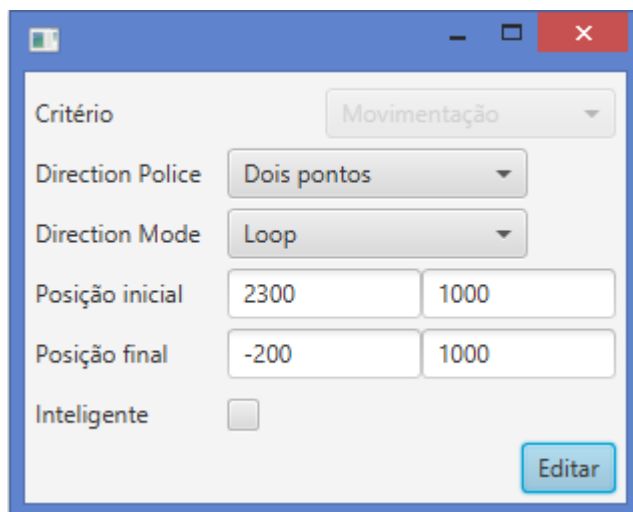


Figura 17 - Configuração Movimentação Inimigo.

1.2.6. Bônus

Bônus são objetos que ao colidir com o ator principal geram algum benefício. Suas características específicas são *score*, destruível, *lifetime* e políticas. O *score* define quanto de benefício será dado ao ator quando houver colisão. *Lifetime* é o tempo de vida do bônus. Política é o que o bônus irá fornecer de benefício para o ator, no caso do exemplo abaixo, irá fornecer *score*.

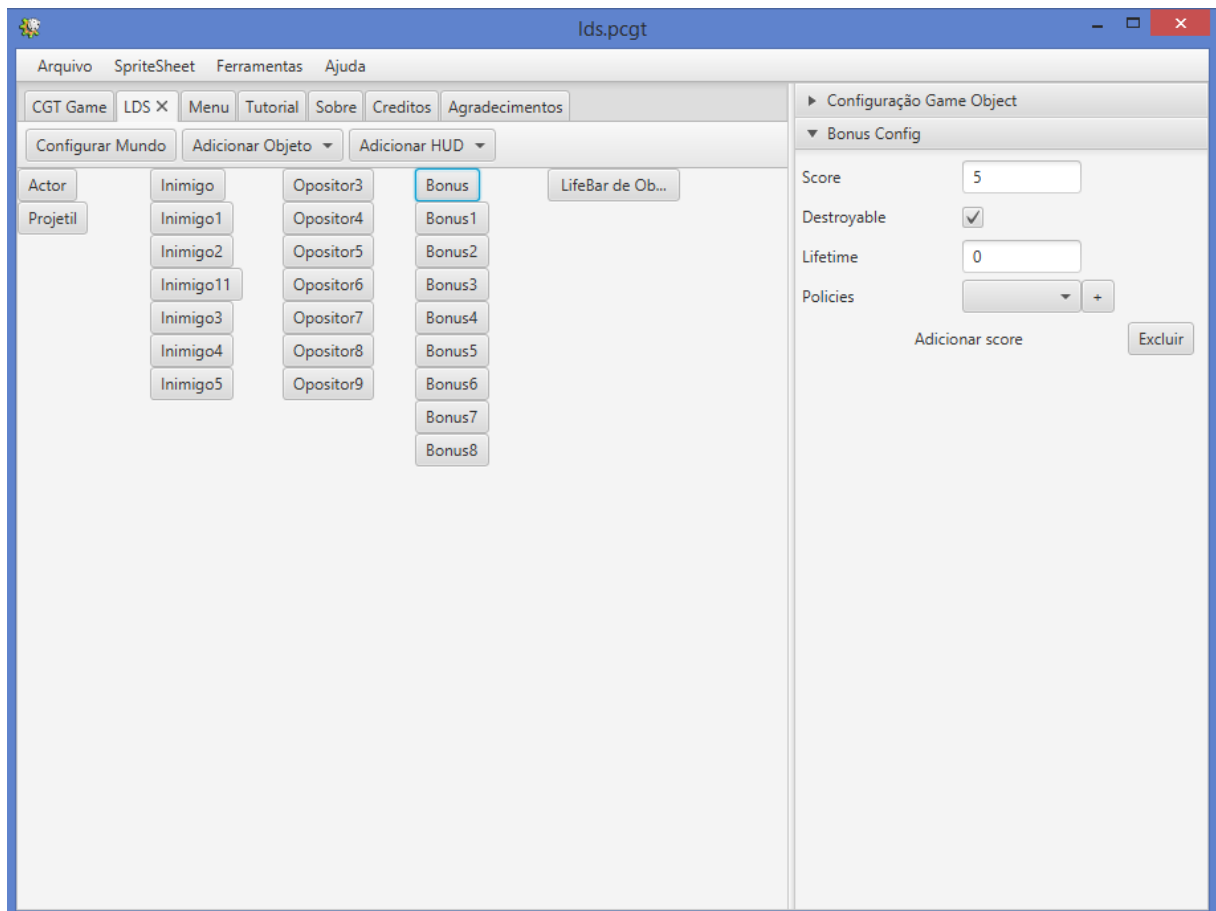


Figura 18 - Configuração Específica Bônus.

1.2.7. Configurar Projétil

O projétil é uma arma que o ator possui, nela é possível configurar que o ator possa destruir inimigos. As configurações específicas são grupo, dano, intervalo, ângulo e posicionamentos. O grupo define a que grupo de inimigos aquele projétil irá afetar. Dano é o valor que irá retirar do inimigo. Posicionamentos são as posições iniciais do projétil referente ao estado do ator.

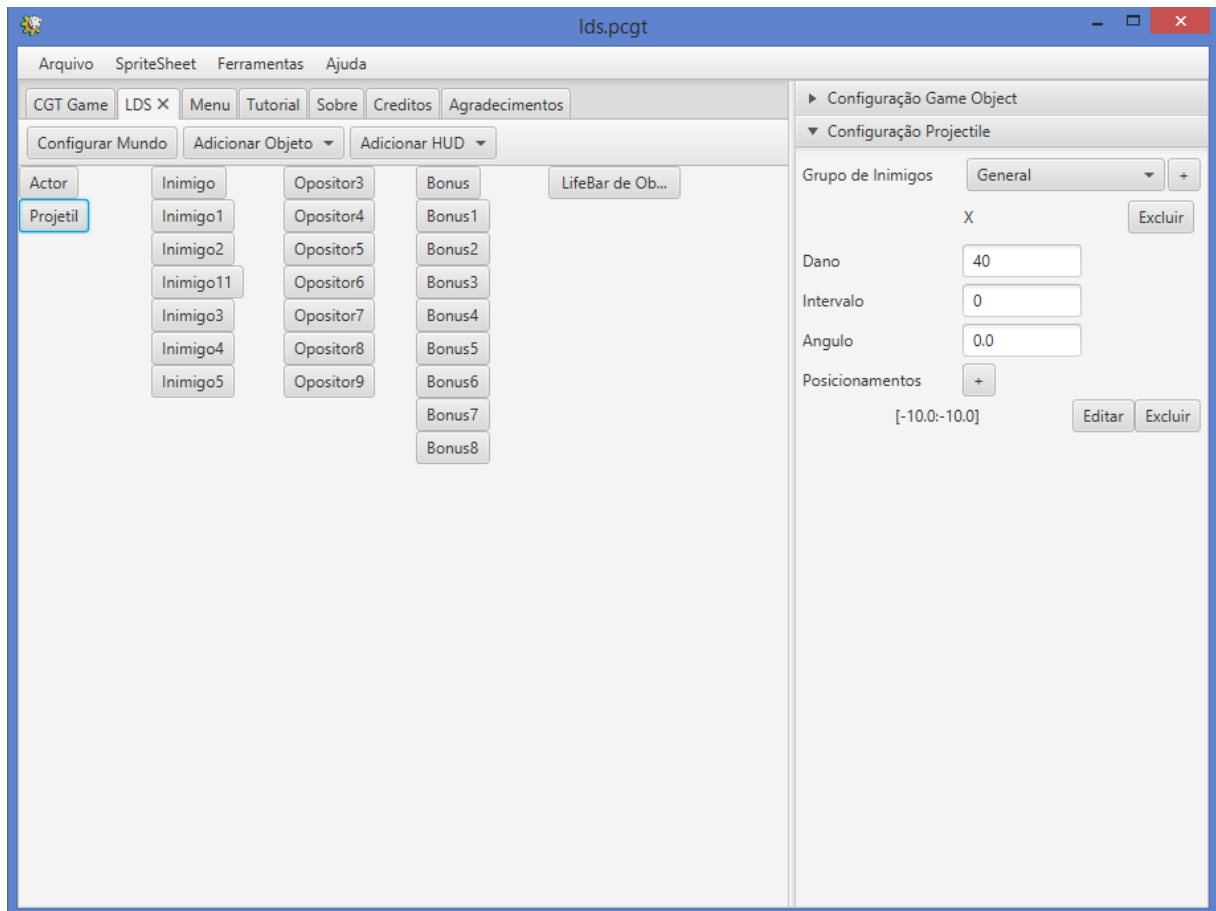


Figura 19 - Configuração Específica Projétil.

1.2.8. Configurar HUD

HUD são elementos de informação que irão ser exibidos na tela, por exemplo vida, munição, tempo, *score*, e outros. No jogo criado foi configurado apenas um *lifebar* mostrando a vida do ator no mundo. Em sua configuração é escolhido o objeto do qual se quer exibir a vida, a imagem da barra e imagem de fundo, além do posicionamento e dimensão que a barra vai possuir.

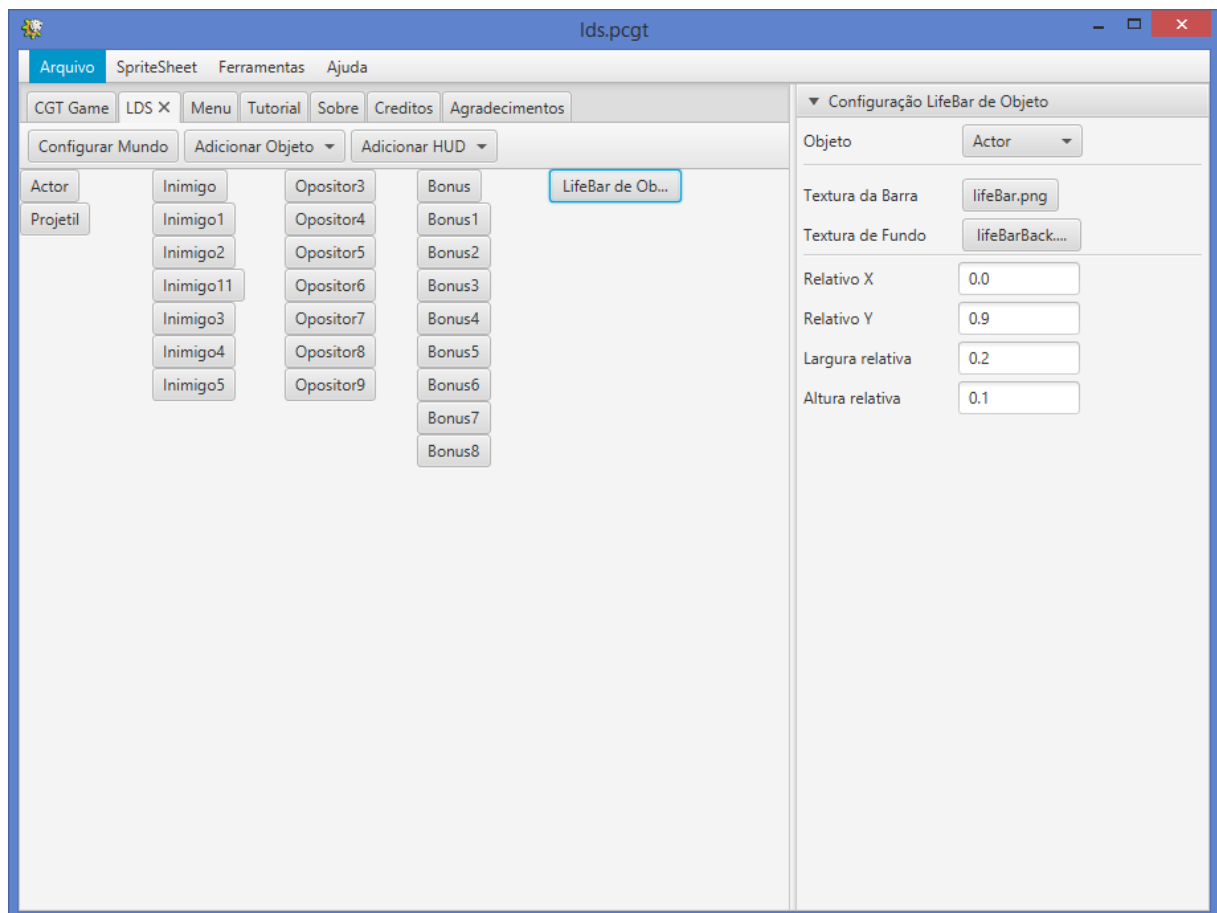


Figura 20 - Configuração *HUD*.

Com o jogo configurado, será mostrado o resultado. Foram demonstradas as configurações de um objeto de cada tipo, porém, percebe-se que foi adicionado uma quantidade bem maior de inimigos e outros objetos vistos. Para adicionar mais objetos, basta seguir as orientações e adicionar quantos forem desejados..



Figura 21 - Configurar Botão na Screen.

2. Tela por tela da ferramenta

2.1. Configurar *Screen*

Criar *Screen*: Para criar uma *screen*, é necessário clicar no botão destacado abaixo e fornecer um nome para a *screen* desejada.

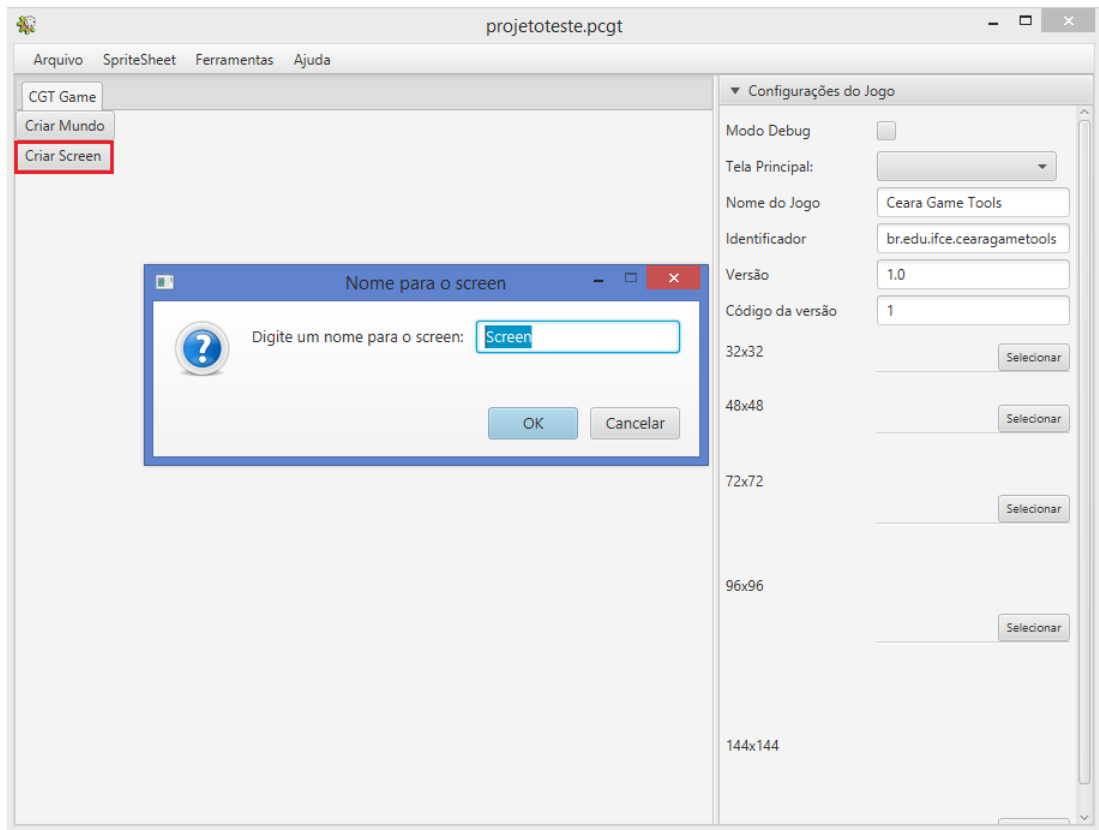


Figura 22 - Criar *Screen*.

Configurar *Screen*: Para configurar a *screen* é preciso clicar na *screen* criada e configurar o *background* que é a imagem do fundo (.png). Outro atributo configurável é o som que a *screen* vai possuir, que deve ser no formato mp3.

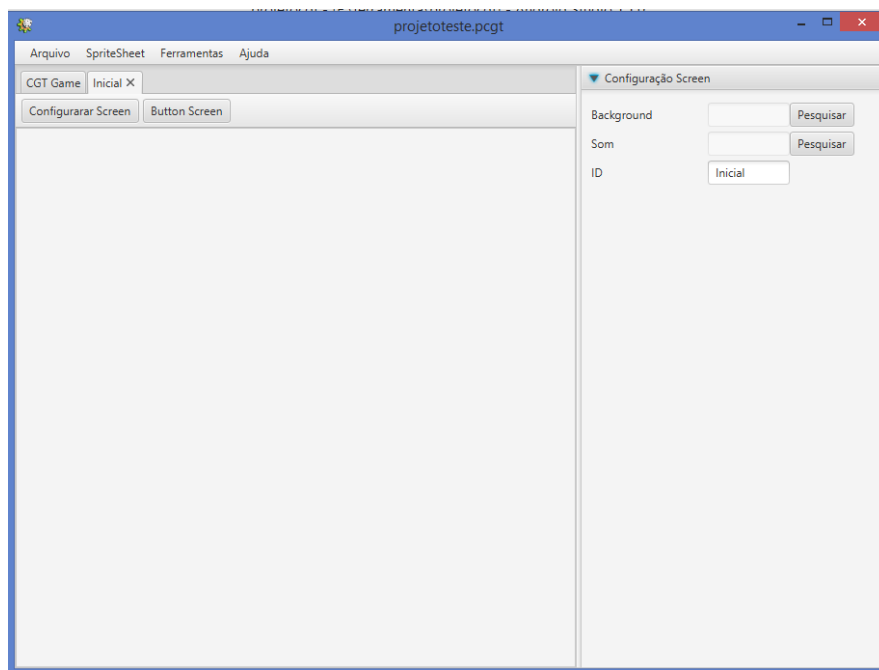


Figura 23 - Configurar Screen.

Adicionar Botões para Screen: Para adicionar um botão a uma *screen* basta clicar no botão em destaque na imagem abaixo. As configurações do botão são as texturas do botão normal e quando for pressionado, a ação dele, que pode ser escolhida no campo “Ir para”, direciona a uma *screen* ou a um mundo, o posicionamento e as suas dimensões na tela com os campos recebendo entradas de 0 a 1 de acordo com a *screen*.

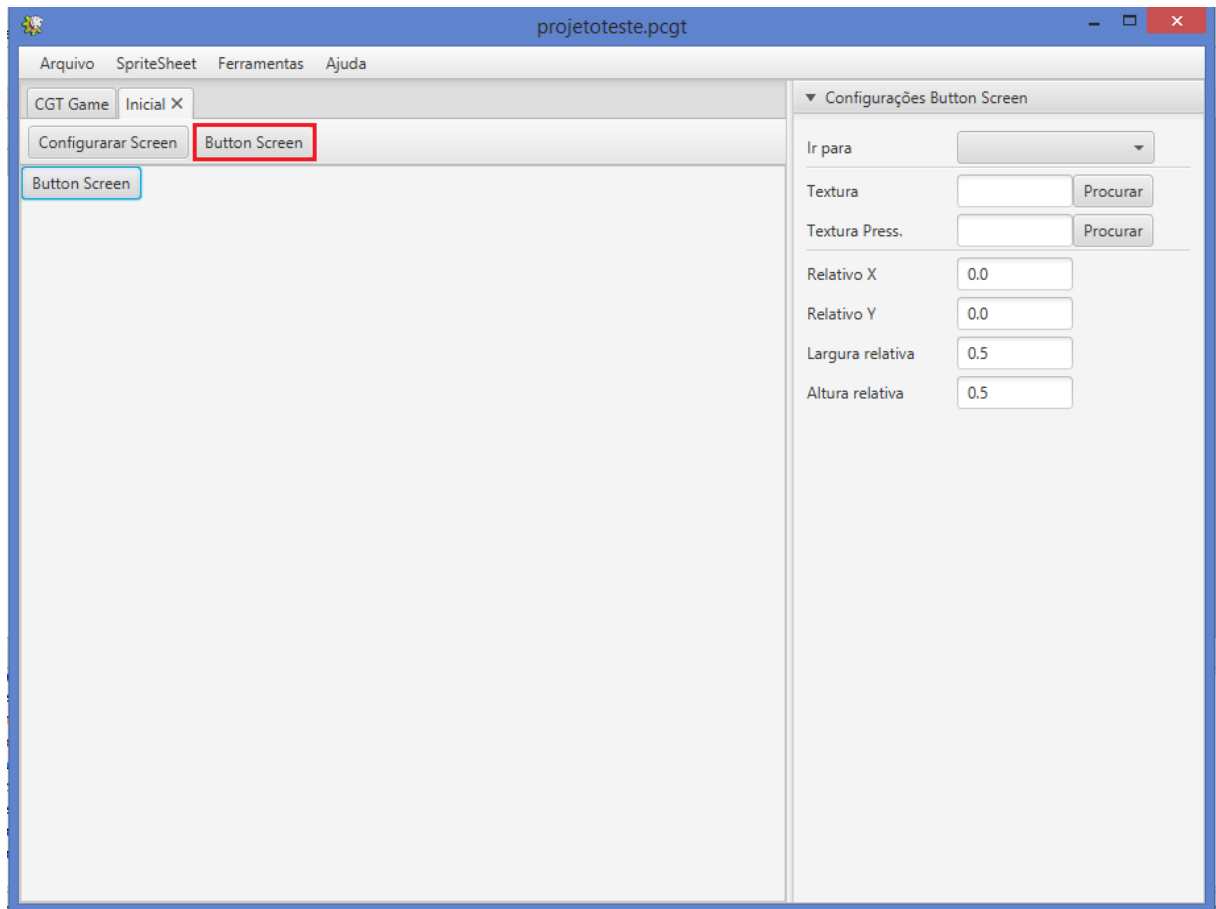


Figura 24 - Configurar Botão na *Screen*.

2.2. Configurar Mundo

2.2.1. Configurações Gerais

Criar Mundo: Para criar o mundo basta clicar no botão destacado na imagem e fornecer o nome para o mundo que se deseja criar.

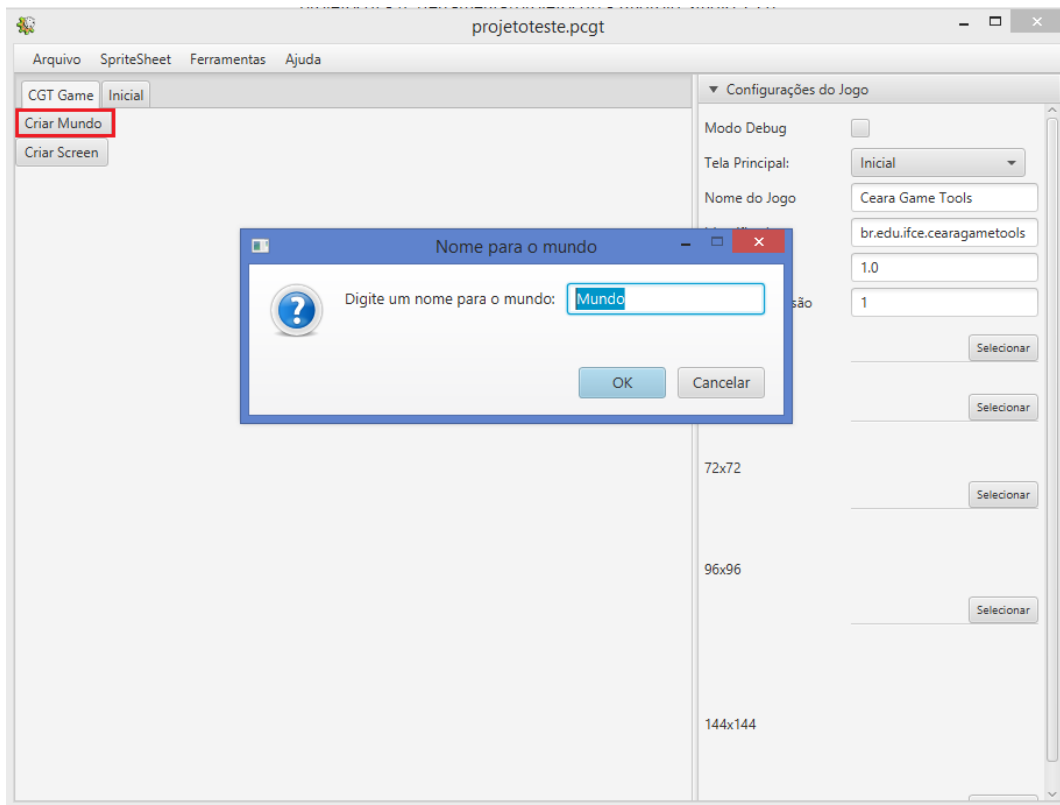


Figura 25 - Criar Mundo.

Configurar Mundo: Para configurar o mundo é preciso selecionar uma imagem para o plano de fundo (.png), configurar *pop-ups*, adicionar critérios de vitória e derrota e a música que o mundo irá possuir.

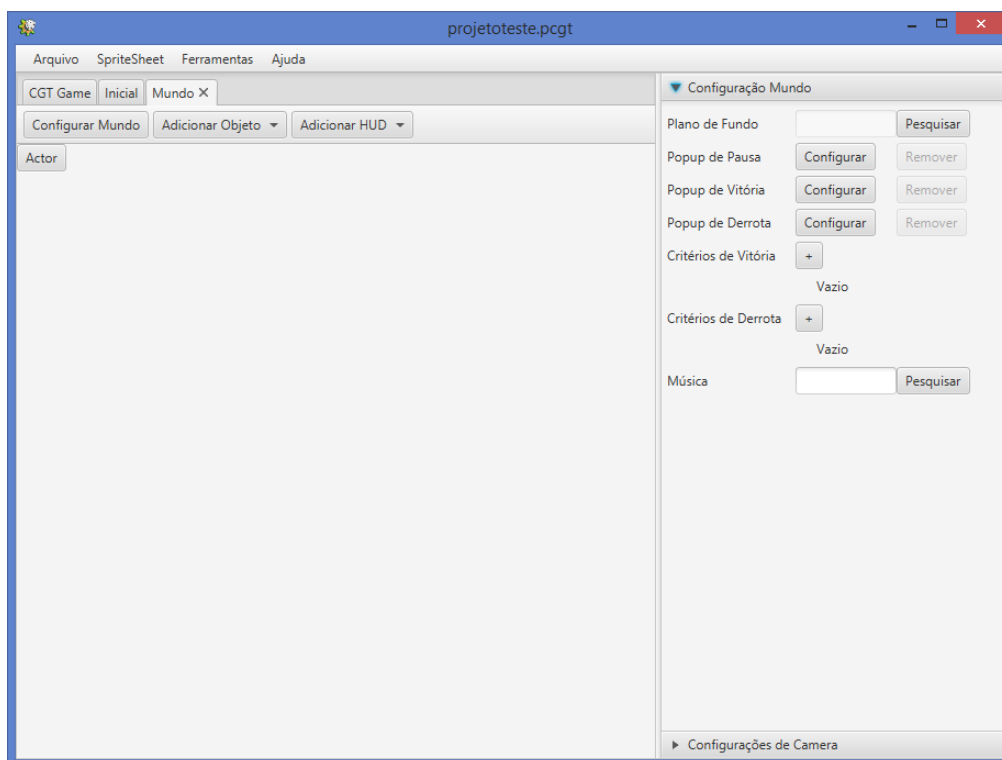


Figura 26 - Configurar Mundo.

2.2.2. Configurar *Pop-up*

Configuração: Para configurar o *Pop-up* o usuário deve selecionar a sua imagem de plano de fundo, a textura da borda horizontal e do canto inferior direito (todas as imagens devem ser .png), além de configurar posicionamento e dimensões na tela.

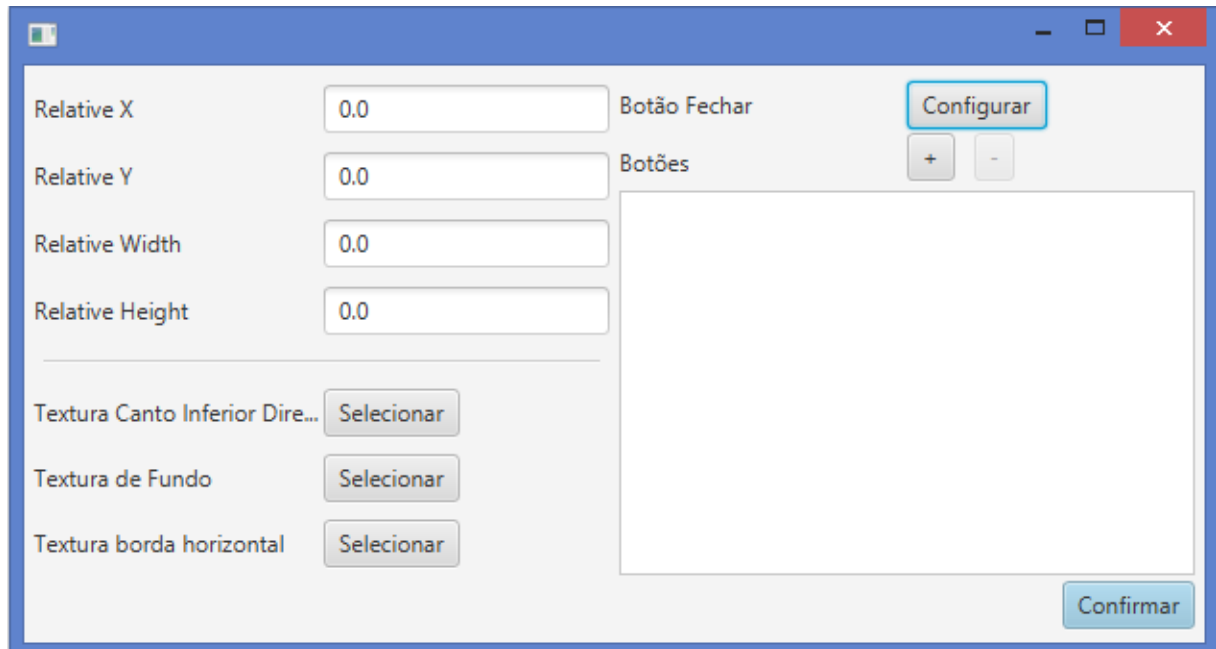


Figura 27 - Configurar *Pop-up*.

Adicionar Botão no *Pop-up*: Para adicionar botões ao *Pop-up* basta clicar no botão destacado na imagem abaixo. As configurações do botão são as texturas do botão em estado normal e quando pressionado, a ação, deve ser escolhida no campo “Ir para”, selecionando a *screen* ou o mundo que a que esse botão vai levar, o posicionamento e as dimensões que ele deve ter na tela com os campos recebendo entradas de 0 a 1 de acordo com a *screen*.

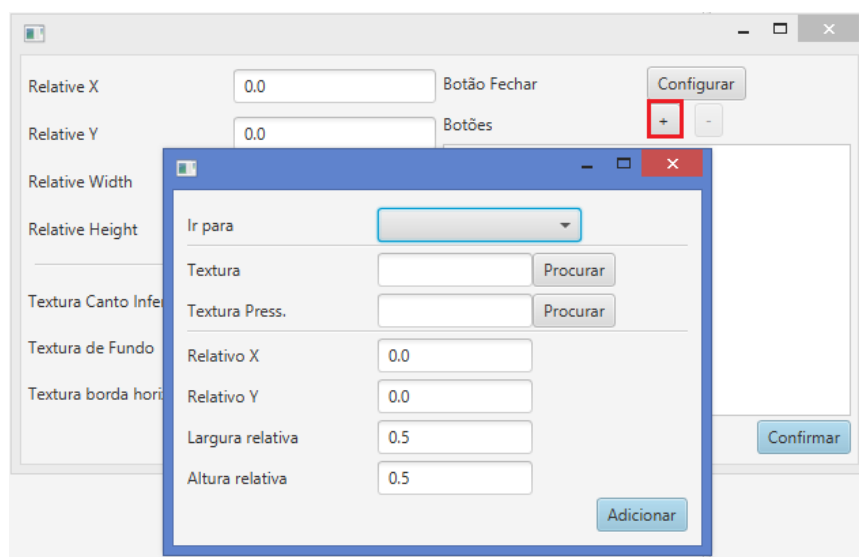


Figura 28 - Adicionar Botao no *Pop-up*.

2.2.3. Configurar critérios de Vitória

Matar Todos os Inimigos: Para configurar o critério de vitória “matar todos os inimigos” basta selecioná-lo e clicar em adicionar. Este critério será atingido quando todos os inimigos que forem destruíveis forem derrotados.

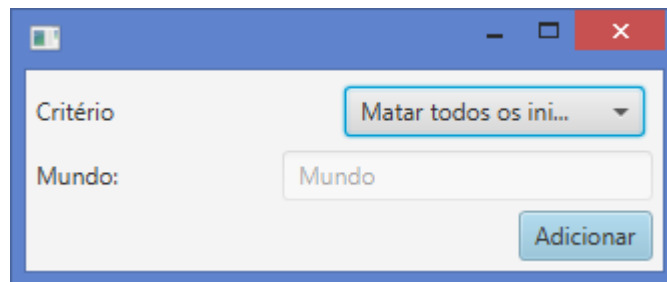


Figura 29 - Adicionar critério de Vitória matar todos os inimigos.

Pegar Todos os Bônus: Para configurar o critério de vitória “pegar todos os bônus” é necessário somente selecioná-lo e clicar em adicionar. Este critério será atingido quando todos os bônus existentes tiverem sido consumidos pelo ator.

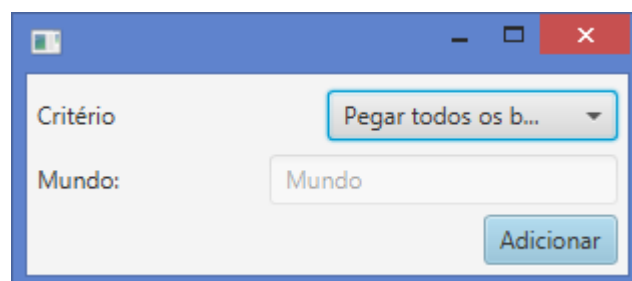


Figura 30 - Adicionar critério de pegar todos os bônus.

Atingir Score: Para configurar o critério de vitória “atingir score” basta selecioná-lo e configurar qual o valor (inteiro) de score deve ser atingido para que o critério seja atingido.

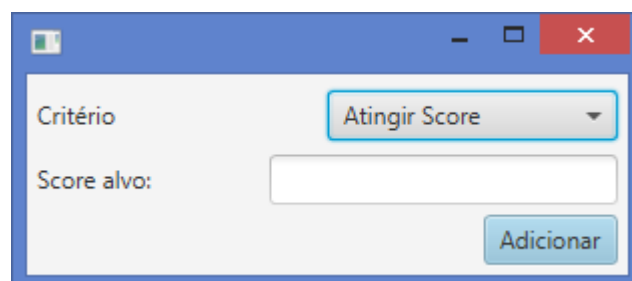


Figura 31 - Adicionar critério de vitória atingir score.

Sobreviver: Para configurar o critério de vitória “sobreviver” é necessário selecioná-lo e informar o tempo (valor inteiro em segundos) que o personagem deve sobreviver para que o critério seja atingido.

A screenshot of a software window with a blue title bar and standard Windows window controls (minimize, maximize, close). The window contains a form with a label 'Critério' next to a dropdown menu showing 'Sobreviver'. Below this is a label 'Tempo:' followed by an empty text input field. At the bottom right of the form is a blue button labeled 'Adicionar'.

Figura 32 - Adicionar critério de vitória sobreviver.

Completar Percurso: Para configurar o critério de vitória “completar percurso” é preciso selecioná-lo e definir o ponto que deve ser alcançado, como também a largura e altura desse ponto, tendo como referência valores inteiros. O critério será atingido quando o personagem alcançar o ponto definido.

 A screenshot of a software window with a blue title bar and standard Windows window controls. The window contains a form with a label 'Critério' next to a dropdown menu showing 'Completar Percur...'. Below this are four labels: 'X', 'Y', 'Largura', and 'Altura', each followed by an empty text input field. At the bottom right of the form is a blue button labeled 'Adicionar'.

Figura 33 - Adicionar critério de vitória completar percurso.

2.2.4. Critérios de Derrota

Ator morrer: Para configurar o critério de derrota “ator morrer” basta selecioná-lo e clicar em adicionar. Esse critério será atingido quando o personagem possuir vida igual ou menor que zero.

 A screenshot of a software window with a blue title bar and standard Windows window controls. The window contains a form with a label 'Critério' next to a dropdown menu showing 'Ator morrer'. Below this is a label 'Ator' followed by a text input field containing the word 'Actor'. At the bottom right of the form is a blue button labeled 'Adicionar'.

Figura 33 - Adicionar critério de Derrota ator morrer.

Contagem Regressiva: Para configurar o critério de “contagem regressiva” necessita selecioná-lo e definir o tempo (em segundos), tamanho e cor da fonte que será exibida na tela e o posicionamento desse tempo na tela. Este critério será cumprido se não for atingido nenhum critério de vitória ao término do tempo.

Critério	Contagem regres...
Tempo	10
Texto	10
Tamanho	10
Cor	255
Relativo X	0.45
Relativo Y	0.9
Largura relativa	0.5
Altura relativa	0.05

Adicionar

Figura 34 - Adicionar critério de derrota contagem regressiva.

2.2.5. Configurar *Spritesheet*

Para configurar *spritesheet* basta clicar em “*spritesheet*” e clicar em adicionar. O usuário deve selecionar o *spritesheet*(.png) e informar o número de linhas e colunas que a imagem possui.

Selecione uma image

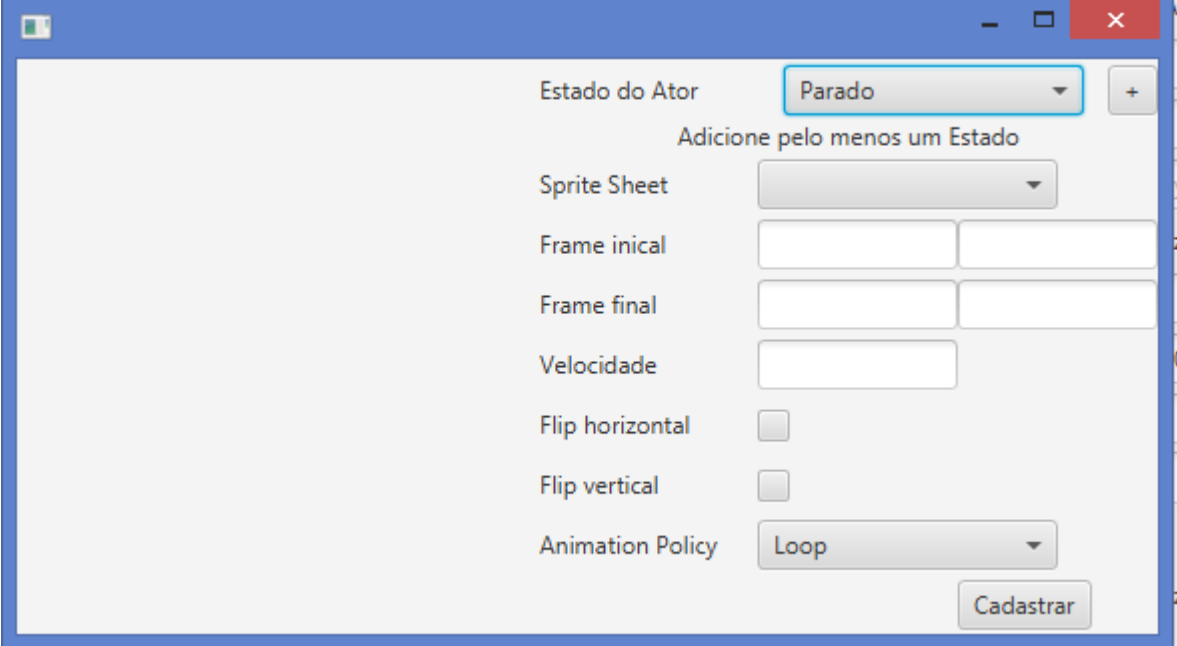
SpriteSheet		Procurar
Nome		
Número de linhas		
Número de colunas		

Confirmar

Figura 35 - Adicionar Spritesheet.

2.2.6. Configurar Animação(Ator,Inimigo,Opositor,Bonus,Projétíl)

Para configurar a animação o usuário deve adicionar os estados do ator à animação configurada, selecionar o *spritesheet*, informar a imagem inicial e final(indexado a partir do zero) e a política de animação. O *flip* horizontal e vertical vai fazer com que animação ocorra nas imagens invertidas de 180°.



The image shows a software configuration window titled "Adicionar animação". It features a large empty rectangular area on the left for a visual preview. On the right, there are several configuration fields: "Estado do Ator" with a dropdown menu set to "Parado" and a "+" button; "Adicione pelo menos um Estado" text; "Sprite Sheet" with a dropdown menu; "Frame inicial" and "Frame final" with two input fields each; "Velocidade" with a single input field; "Flip horizontal" and "Flip vertical" with checkboxes; and "Animation Policy" with a dropdown menu set to "Loop". A "Cadastrar" button is located at the bottom right of the configuration area.

Figura 36 : Adicionar animação.

2.2.7. Configuração Comum dos Objetos(Ator,Inimigo,Opositor,Bonus,Projétíl)

Neste tipo de configuração comum dos objetos, adiciona-se animação, determina vida máxima do objeto, vida inicial do objeto, posição inicial, dimensões do objeto(*pixels*), caixa de colisão, velocidade do objeto(*pixels*), os áudios do objeto na tela, quando atingido ou destruído.

Configuração Game Object

Animações +

Nenhuma Animação

Máxima vida

Vida

Posições Iniciais +

Vazio

Dimensões

Colisão

Velocidade

Som Pesquisar

Som de colisão +

Vazio

Som de morte +

Vazio

2.3. Configurações Específicas do Ator

Nas configurações específicas do ator são adicionadas ações, um projétil padrão, caso tenha, e determinado o tempo de recuperação que é o tempo que ele fica “invencível” ao colidir com algum objeto que o cause dano.

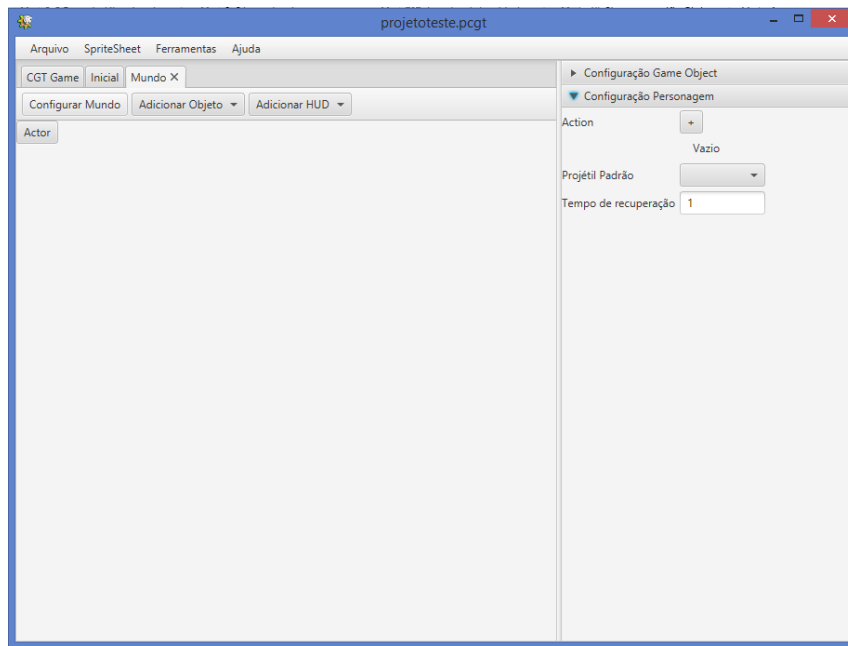


Figura 38 - Configuração específica do ator.

2.3.1. Configuração das actions do ator

Nestas configurações, o usuário deve associar uma entrada a um tipo de movimentação, por exemplo, associar o *Slice Left* a “Andar para esquerda” é configurar que ao deslizar o dedo para esquerda o ator vai andar para esquerda.

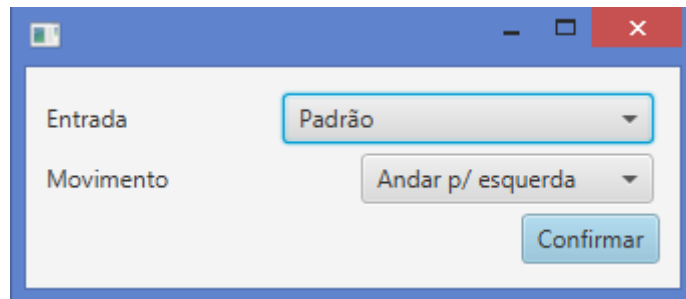


Figura 39 - Adicionando action ao ator.

2.4. Configurações Específicas dos Inimigos

Nas configurações do inimigo, o usuário deve marcar *block* caso queira que o ator ao colidir seja bloqueado, destruível caso o ator possa destruir com o projétil. Além disso, é possível adicionar comportamento, valor do dano que ele causa e o grupo a que ele pertence.

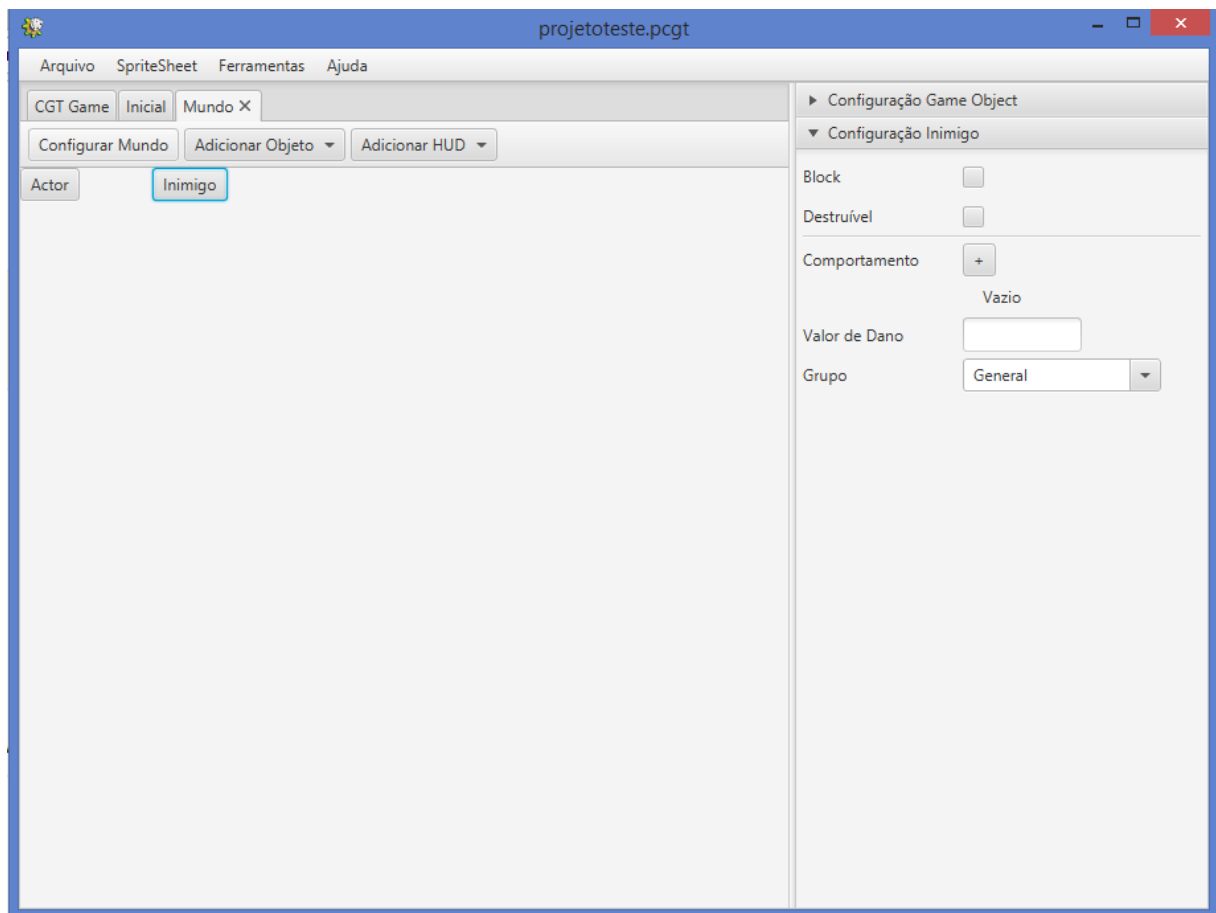


Figura 40 - Configuração específica do inimigo.

2.4.1. Configuração Comportamento Sino

Na configuração do Sino o usuário deve selecionar se o movimento vai ser na largura ou na altura, os valores de mínimo e máximo do movimento.

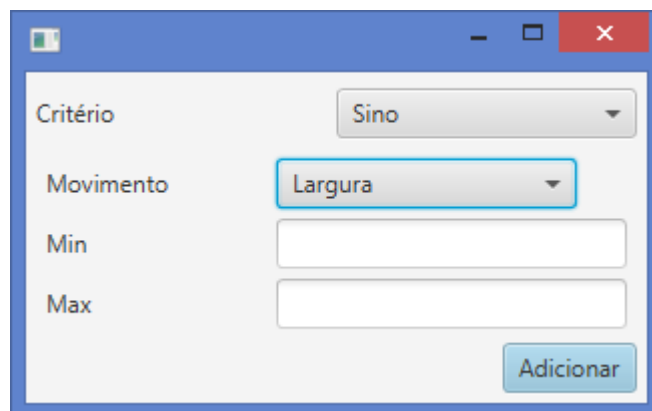


Figura 41 - Configurar movimentação sino.

2.4.2. Configuração Comportamento Fade

Nesta configuração o usuário deve inserir o tempo (segundos) e a política, que pode ser esperar o tempo e o inimigo aparecer ou desaparecer da tela quando o tempo estiver terminado.

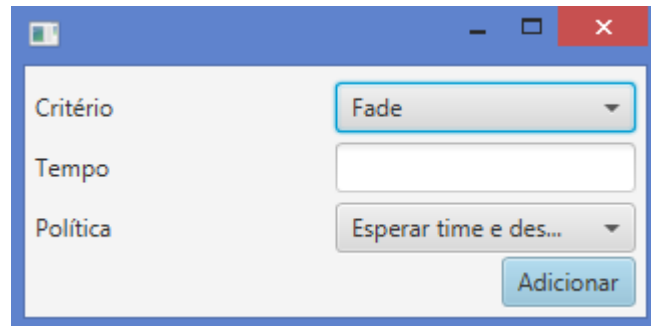
A screenshot of a software window titled 'Configurarar movimentação fade'. The window has a blue title bar with standard Windows window controls (minimize, maximize, close). The main area is light gray and contains three labels on the left: 'Critério', 'Tempo', and 'Política'. To the right of 'Critério' is a dropdown menu showing 'Fade'. To the right of 'Tempo' is an empty text input field. To the right of 'Política' is a dropdown menu showing 'Esperar time e des...'. At the bottom right of the window is a blue button labeled 'Adicionar'.

Figura 42 - Configurarar movimentação *fade*.

2.4.3. Configuração Comportamento Onda

Neste tipo de configuração o usuário deve selecionar a amplitude do movimento, frequência, a fase e o máximo x, que é o limite do movimento no eixo X.

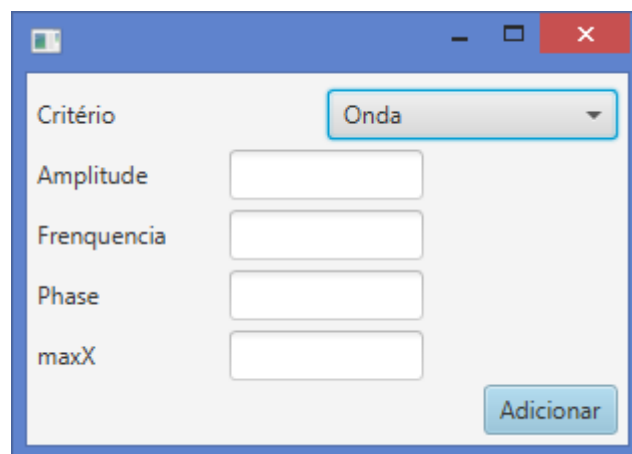
A screenshot of a software window titled 'Configurarar movimentação onda'. The window has a blue title bar with standard Windows window controls (minimize, maximize, close). The main area is light gray and contains five labels on the left: 'Critério', 'Amplitude', 'Frenquencia', 'Phase', and 'maxX'. To the right of 'Critério' is a dropdown menu showing 'Onda'. To the right of each of the other four labels is an empty text input field. At the bottom right of the window is a blue button labeled 'Adicionar'.

Figura 43 - Configurarar movimentação onda.

2.5. Configurações Específicas dos Opositores

A configuração específica do opositor determina o tipo do objeto, ou seja, se ele será bloqueante para o ator e se poderá ser destruível.

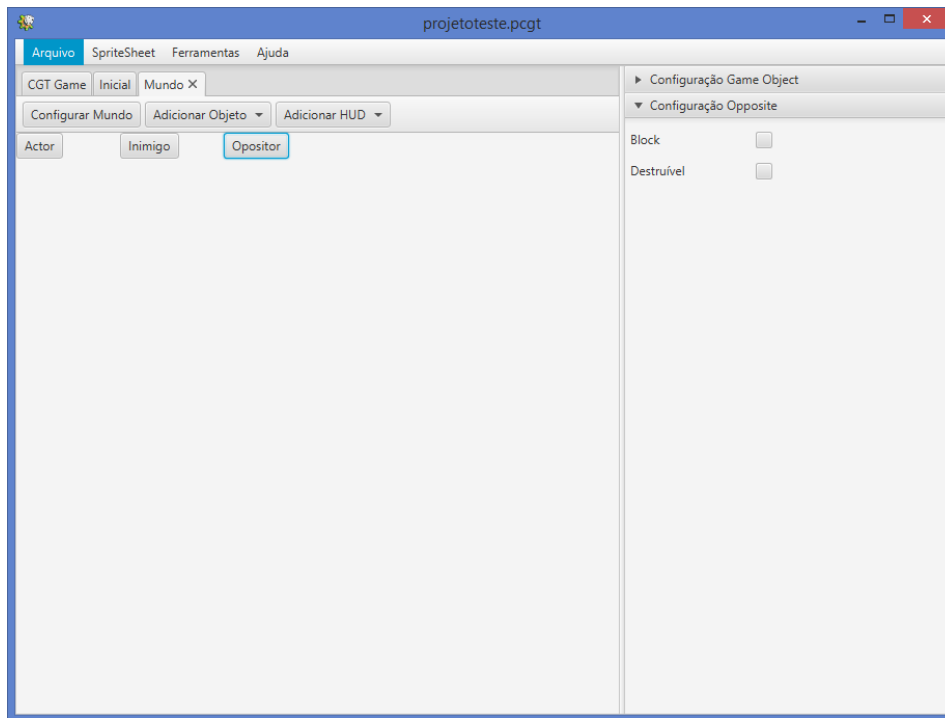


Figura 44 - Configuração específica opositor.

2.6. Configurações Específicas dos Bônus

Nesta configuração, o usuário determina o score, ou seja, o valor de recarga ao colidir com o ator. Além disso, se o bônus é destruível, o tempo de vida e a política, que pode ser de adicionar vida, adicionar score ou adicionar bala para sua arma.

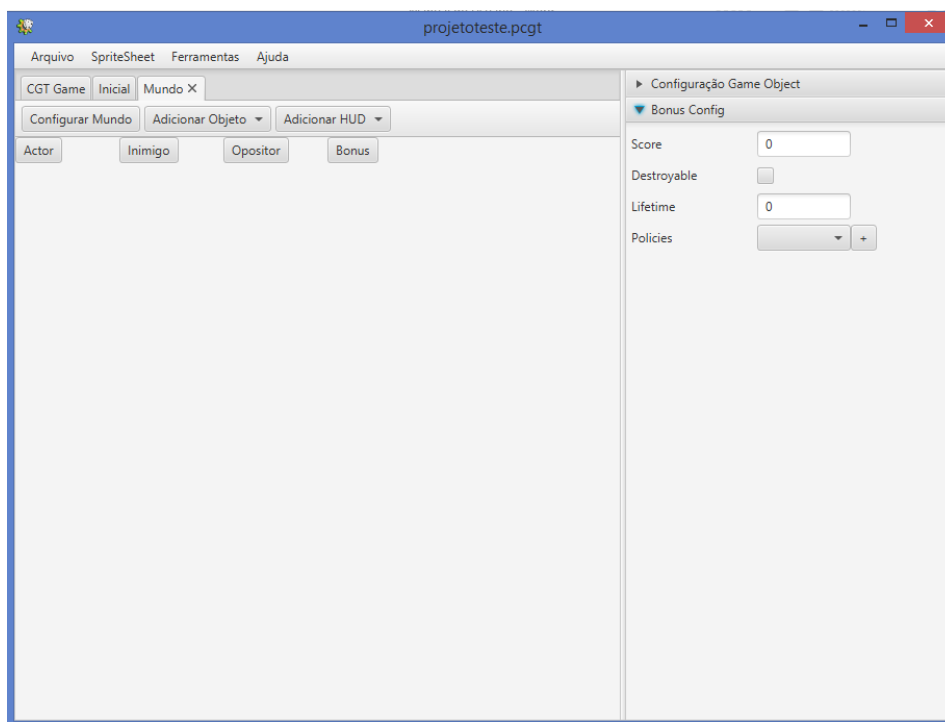


Figura 45 - Configuração específica bônus.

2.7. Configurar Projétil

Na configuração do projétil o usuário deve selecionar qual o grupo de inimigos que o projétil vai afetar, dano do projétil, intervalo de tempo entre os tiros (segundos), ângulo e ajuste de posicionamento.

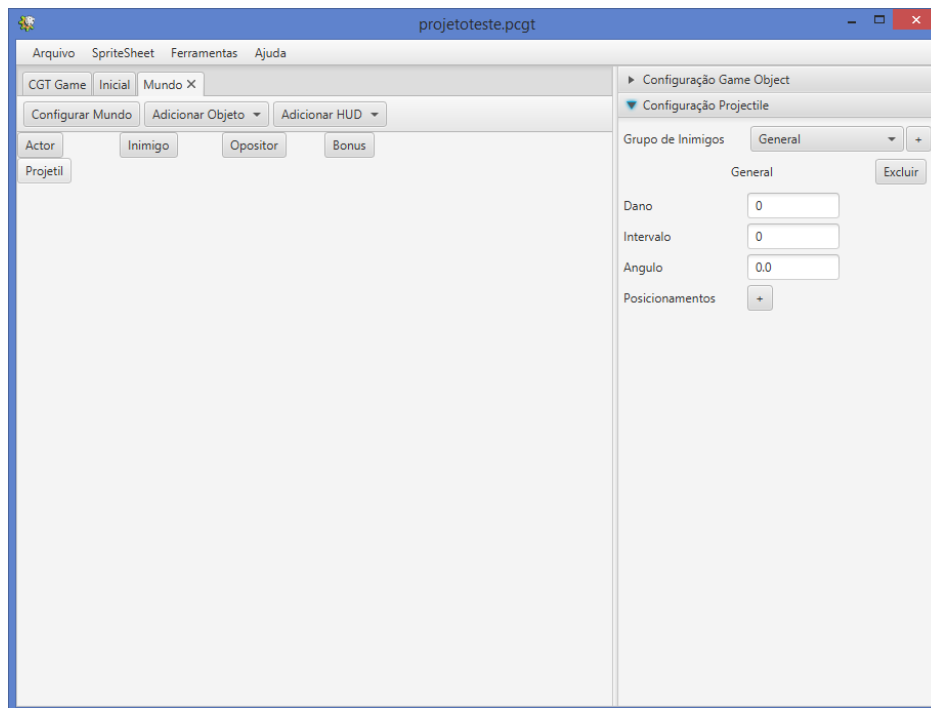


Figura 46 - Configuração específica projétil.

2.7.1. Configuração do Posicionamento

Nesta configuração, o usuário determina o ponto de partida do projétil em relação ao estado do ator. Configura-se a posição relativa ao ator e o estado associado.

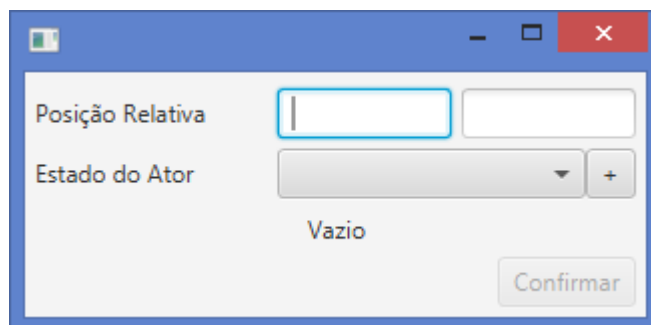


Figura 47 - Configurarar posicionamento projéti.

2.8. Configurar HUDS

2.8.1. Configuração do *Display* de Munição

Esta configuração é através do ajuste do posicionamento na tela, selecionar uma imagem e o projétil do qual o usuário quer visualizar a munição.

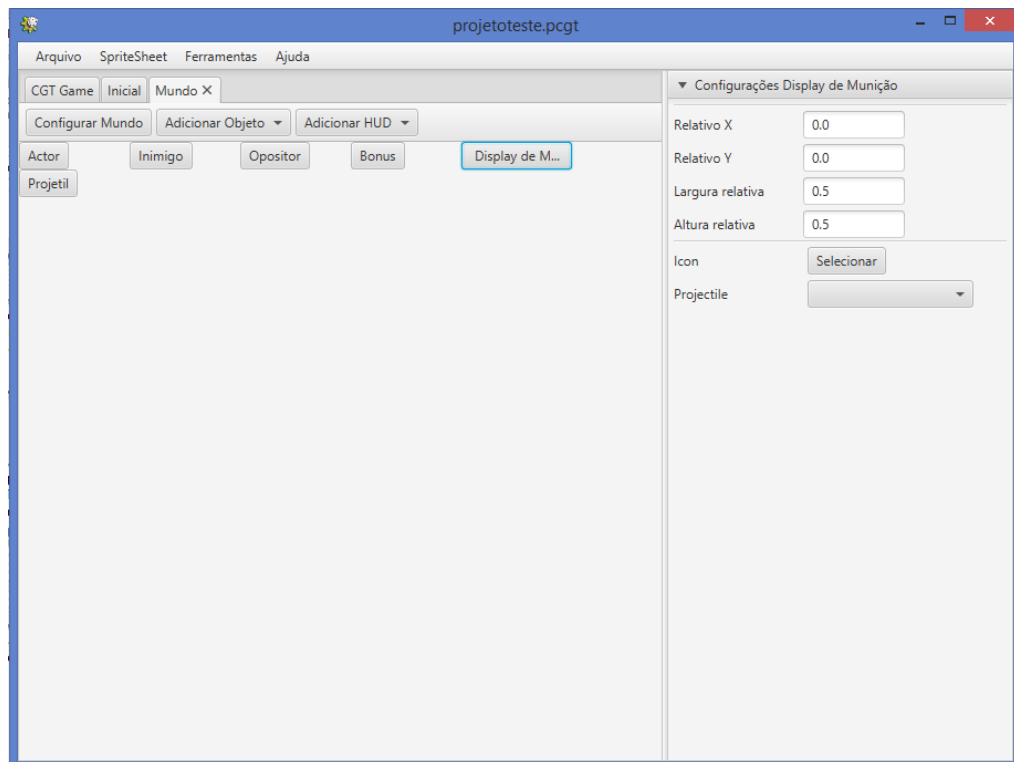


Figura 48 - Configuração do display de munição.

2.8.2. Configuração do *LifeBar* Inimigos

Este tipo de configuração mostra um *LifeBar* com os inimigos destruíveis no mundo. Suas definições se resumem ao posicionamento e dimensão do *LifeBar* e textura de fundo e a textura da barra que devem ser .png.

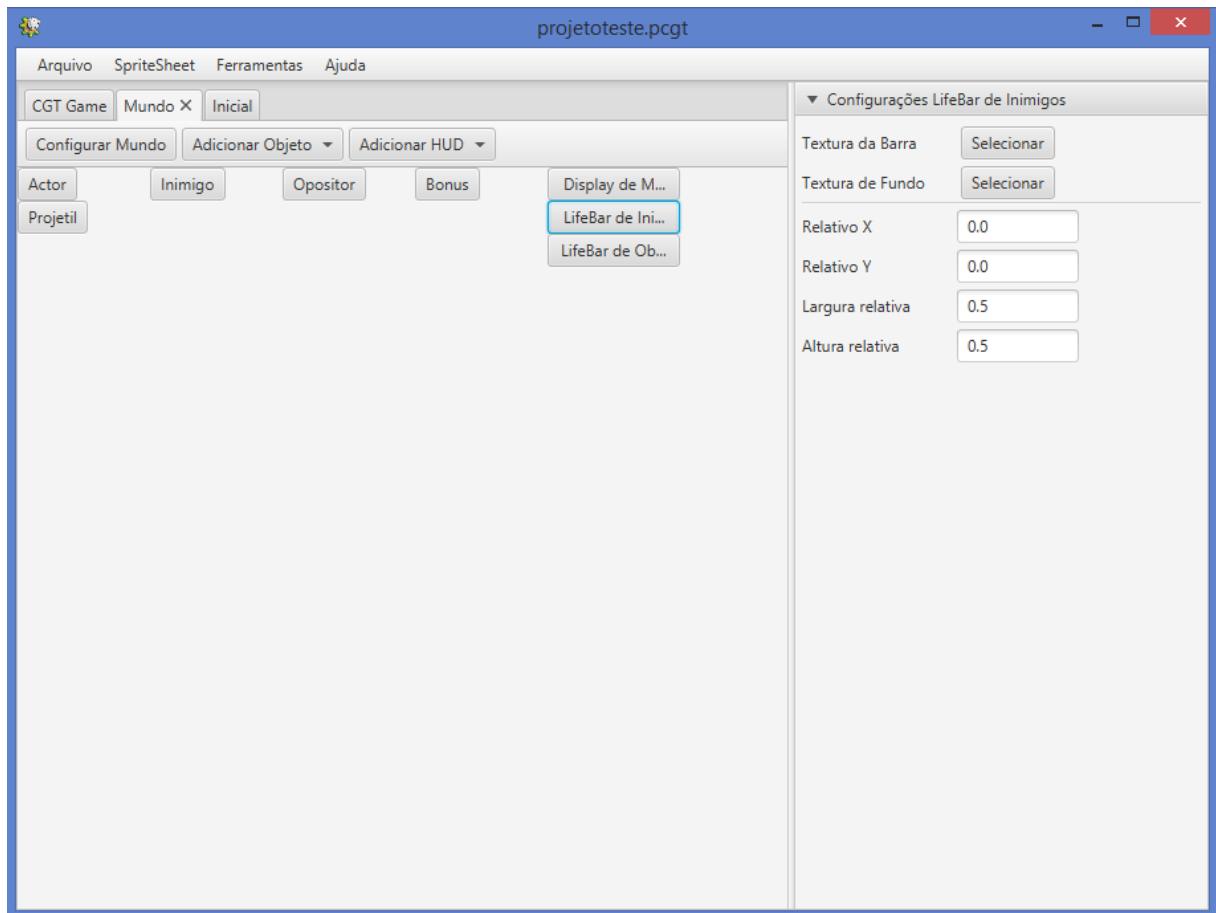


Figura 49 - Configuração do lifebar inimigos.

2.8.3. Configuração *LifeBar* Objeto

Esta configuração mostra um *LifeBar* com o life do objeto escolhido. Deve-se selecionar textura de fundo e da barra, objeto que a barra se refere e posicionamento e dimensões na tela.

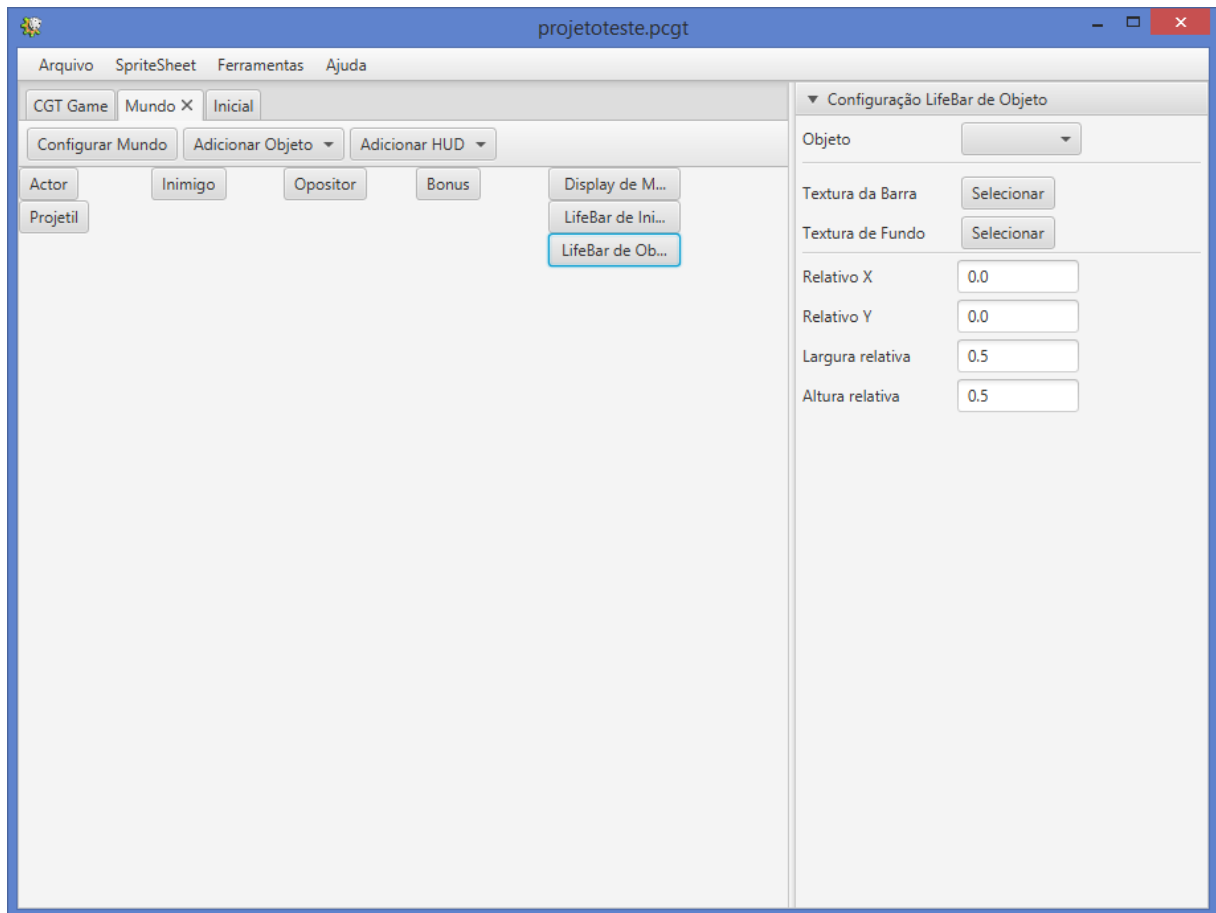


Figura 50 - Configuração *lifebar* objetos.

3. Executar projeto

Para executar o jogo que está sendo criado, basta clicar em “ferramentas” e depois clicar em “rodar”.

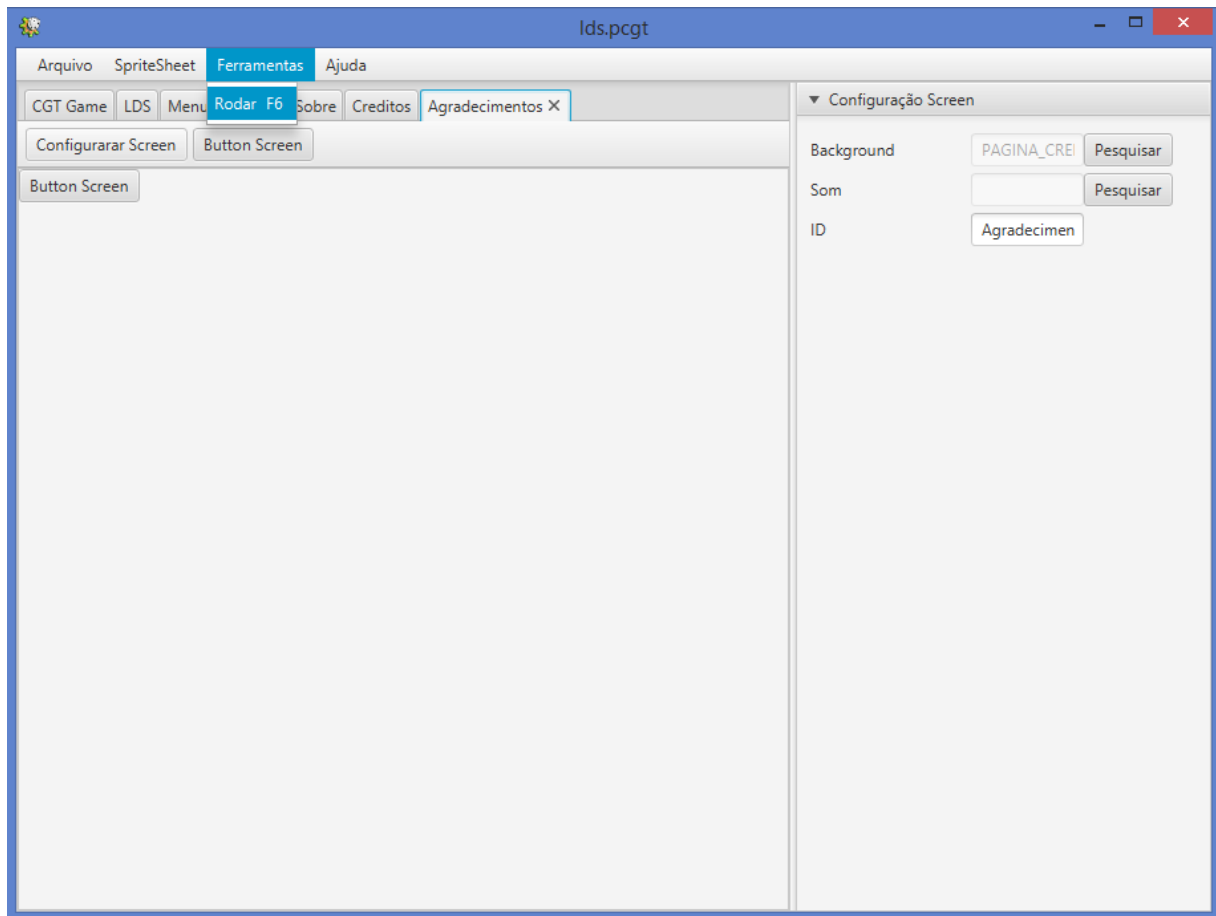


Figura 51 - Executar jogo.

4. Configurar *Game*

Nesta configuração o usuário poderá configurar *o mode debug* que habilita a visualização das caixas de colisão dos objetos, que ajuda na configuração do mundo, configurar a tela inicial do jogo, selecionando uma *screen* ou um mundo e selecionar as imagens que serão os ícones do jogo criado..

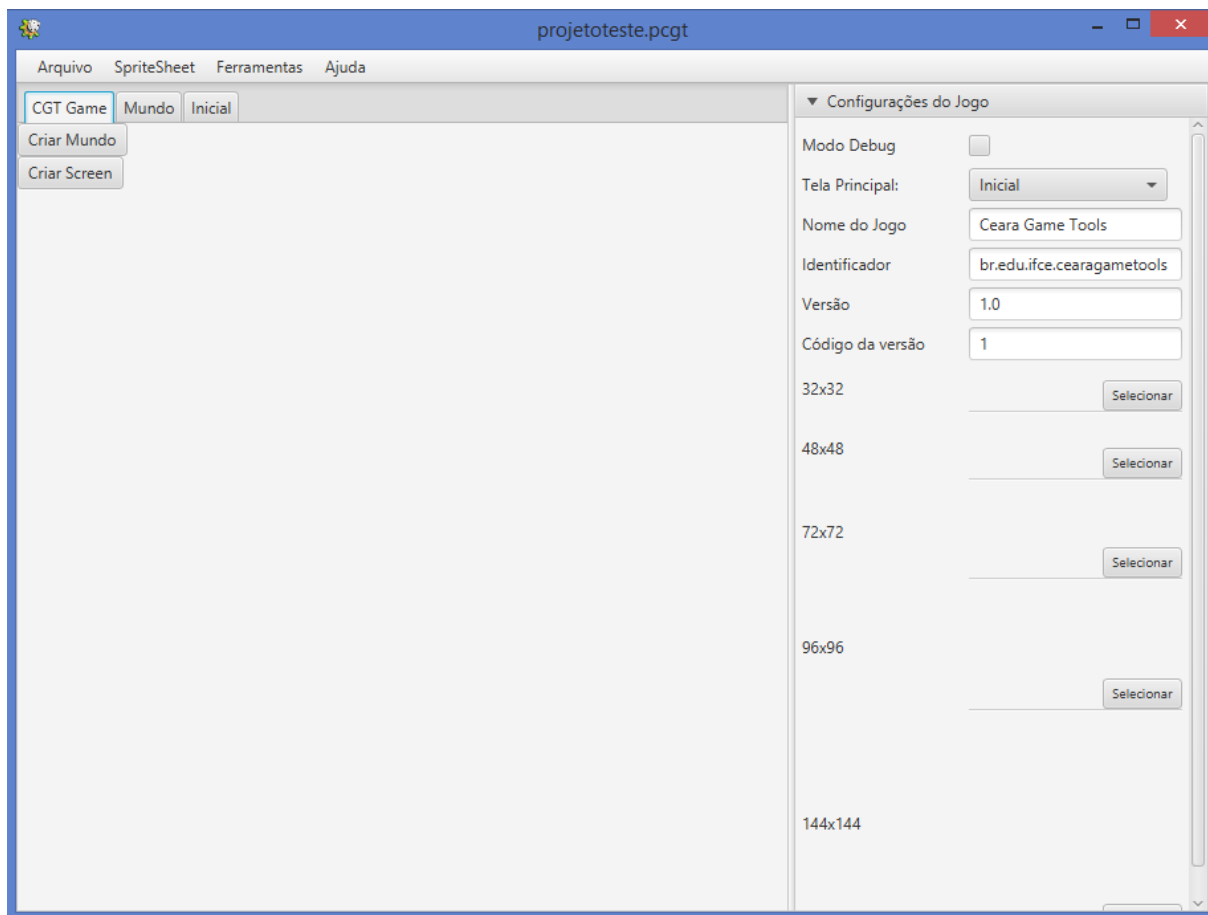


Figura 52 - Configuração do *Game*.

5. Exportar projeto

Para exportar o projeto para Android, é preciso clicar em “arquivo” e “exportar”. Depois é necessário informar a pasta do *sdk*, as versões que se deseja do Android e a chave de assinatura para poder gerar o .apk.

The image shows a software window with a blue title bar and standard Windows window controls (minimize, maximize, close). The window contains a form with the following elements:

- Android SDK:** A text input field followed by a "Procurar" button.
- Versão alvo:** A text input field.
- Versão mínima:** A text input field.
- Versão Build Tools:** A text input field.
- Chave de assinatura:** A text input field followed by a "Procurar" button.
- Senha:** A text input field.
- Apelido da chave:** A text input field.
- Senha:** A text input field.
- Buttons:** An "Iniciar" button (top right), a "Criar Chave" button (middle right), and a "Detalhes" button (bottom left).

Figura 53 - Exportar jogo para android.

6. Gerar Chave de Assinatura

A geração da chave de assinatura é feita através de um programa chamado portecle, mas que já vem integrado a ferramenta. Para criar, basta clicar em “Criar Chave” como mostra na Figura 53. O programa irá ser inicializado e o próximo passo é a criação da chave.

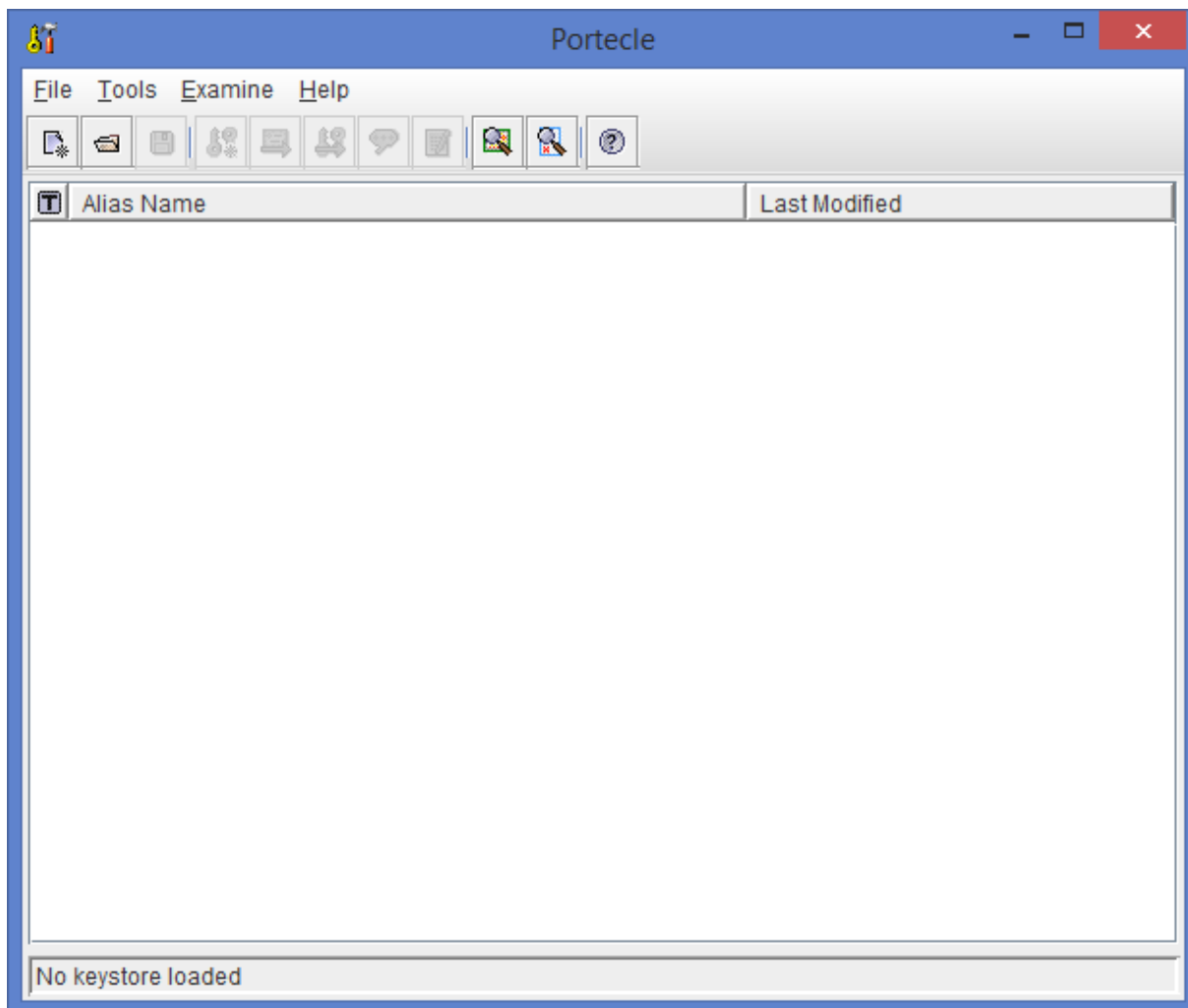


Figura 54 - Software Portecle.

Para criar a chave basta clicar em *File -> new keystore*. O tipo de *keystore* é o *default* JKS.

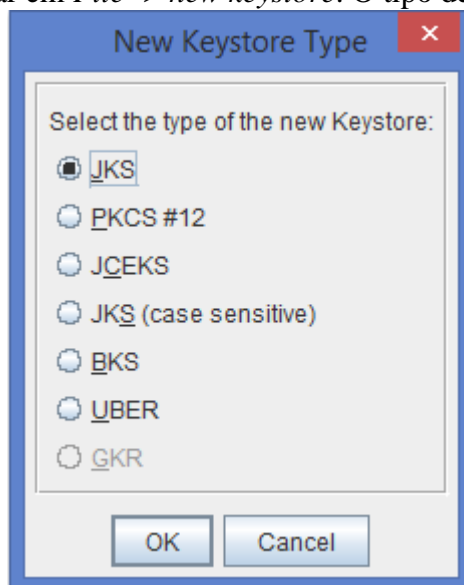


Figura 55 - Tela de criação de chave.

Depois clique em *Tools -> Generate Key Pair*. O algoritmo é o *default* e basta clicar em *OK*. Na Figura 57 usuário deve informar os dados pedidos e depois salvar a chave criada.

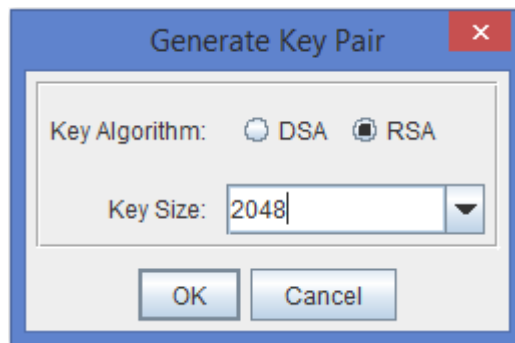


Figura 56 : Primeira tela Generate Key Pair.

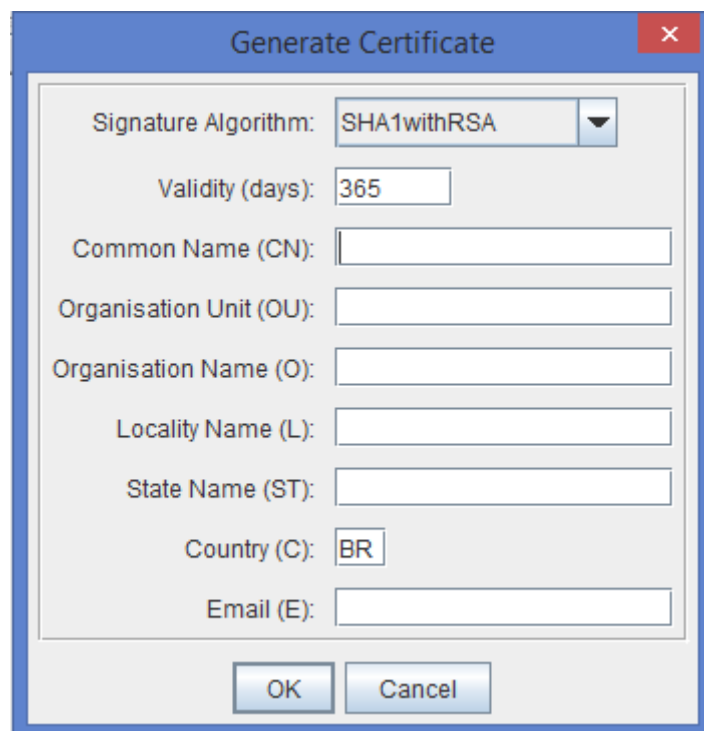


Figura 57 - Primeira tela Generate Key Pair.