

Joel Xavier Rocha

**Módulo de pré visualização para a ferramenta  
CGT - Ceará Game Tools**

Fortaleza

2015

Joel Xavier Rocha

## **Módulo de pré visualização para a ferramenta CGT - Ceará Game Tools**

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.

Instituto Federal de Ciência, Educação e Tecnologia do Ceará

Engenharia de Computação

Orientador: Prof. Dr. Carlos Hairon Ribeiro Gonçalves

Fortaleza

2015

# Resumo

O crescimento da indústria de jogos em diversas áreas - como entretenimento e educação - torna necessário a elaboração de ferramentas que possibilitem a criação destes de forma simples, fácil e objetiva. Por esta razão, o projeto *Ceará Game Tools* (CGT), tem como objetivo possibilitar isso aos usuários que não conhecem os meios e as técnicas envolvidas na criação de jogos.

Então, é proposto nesse trabalho, melhorias importantes para a ferramenta CGT, principalmente, a pré visualização do jogo que está sendo construído e também dos seus objetos.

**Palavras-chave:** CGT, ferramenta;

# Lista de ilustrações

Figura 1 – Problema de pré-visualização na ferramenta 1.0. . . . .	11
Figura 2 – Problema com os controles na configuração de uma animação na ferramenta 1.0. . . . .	11
Figura 3 – Disposição dos objetos na versão anterior da ferramenta. . . . .	16
Figura 4 – Configuração da posição inicial de um objeto na primeira versão da ferramenta. . . . .	17
Figura 5 – Configuração das dimensões de um objeto na primeira versão da ferramenta. . . . .	17
Figura 6 – Painel de configuração de um mundo do jogo na primeira versão da ferramenta. . . . .	18
Figura 7 – Painel de configuração de um objeto do jogo, existe para os seguintes objetos: ator, inimigo, bônus, opositor e projétil. . . . .	19
Figura 8 – Painel de configuração de um botão na tela do jogo. . . . .	19
Figura 9 – Painel de configuração do mostrador de munição de um projétil do jogo. . . . .	19
Figura 10 – Painel de configuração da barra de vida de um objeto do jogo. . . .	20
Figura 11 – Nova tela inicial da ferramenta com os novos controles que possibi- lizaram as melhorias. . . . .	23
Figura 12 – Árvore de objetos da nova versão da ferramenta. . . . .	24
Figura 13 – Painel de configuração de um ator no jogo . . . . .	27
Figura 14 – Janela para adicionar animações para um objeto na primeira versão da ferramenta. . . . .	27
Figura 15 – Painel de configuração das animações do objeto na nova versão da ferramenta. . . . .	28
Figura 16 – Tela da ferramenta mostrando a área de pré visualização com dois objetos configurados. . . . .	29
Figura 17 – Exemplo de dimensão do objeto sendo percebida no momento que é alterada na configuração . . . . .	31
Figura 18 – Inserindo uma nova tela na ferramenta. . . . .	55

Figura 19 – Tela de um jogo exemplo sendo configurada. . . . .	55
Figura 20 – Configurando um botão em tela de exemplo do jogo. . . . .	56
Figura 21 – Adicionando um mundo na nova versão da ferramenta . . . . .	56
Figura 22 – Propriedades gerais de um mundo. . . . .	56
Figura 23 – Propriedades de vitória do mundo. . . . .	57
Figura 24 – Propriedades de derrota do mundo. . . . .	57
Figura 25 – Propriedades da pausa do mundo. . . . .	57
Figura 26 – Propriedades da câmera do mundo. . . . .	58
Figura 27 – Configurações do critério de vitória sobreviver no jogo. . . . .	58
Figura 28 – Configurações do critério de vitória completar percurso no jogo. . . . .	58
Figura 29 – Configurações do critério de vitória de atingir uma pontuação no jogo. . . . .	59
Figura 30 – Configurações do critério de derrota de contagem regressiva no jogo. . . . .	59
Figura 31 – Janela de configuração de uma mensagem no mundo, seja ela para vitória, derrota ou pausa. . . . .	60
Figura 32 – Configuração dos botões de uma mensagem do mundo. . . . .	60
Figura 33 – Janela para adicionar <i>spritesheets</i> na ferramenta. . . . .	61
Figura 34 – Painel de configuração da animação de um objeto . . . . .	61
Figura 35 – Propriedades de uma animação nas configurações de um objeto. . . . .	62
Figura 36 – Lista de animações de um objeto do jogo . . . . .	62
Figura 37 – Propriedades gerais de um objeto . . . . .	63
Figura 38 – Propriedades de dimensões e colisão de um objeto no jogo. . . . .	63
Figura 39 – Configuração dos sons que um objeto faz. . . . .	64
Figura 40 – Propriedades exclusivas do ator do jogo . . . . .	65
Figura 41 – Configurações de um opositor. . . . .	65
Figura 42 – Propriedades do inimigo . . . . .	66
Figura 43 – Propriedades de um bônus do jogo . . . . .	66
Figura 44 – Propriedades de um projétil no jogo. . . . .	67
Figura 45 – Propriedades da barra de vida de um objeto. . . . .	67
Figura 46 – Classes envolvidas no módulo de pré visualização. . . . .	68

# Lista de tabelas

Tabela 1 – Módulos existentes no projeto CGT. . . . .	10
Tabela 2 – Objetos que existem em um jogo. . . . .	14
Tabela 3 – Controles utilizados na primeira versão da ferramenta. . . . .	16
Tabela 4 – Objetos do jogo e suas respectivas propriedades que demandam ser simuladas, ou seja, serem visualizadas na criação. . . . .	18
Tabela 5 – Controles utilizados na segunda versão da ferramenta. . . . .	22
Tabela 6 – Tabela mostrando a hierarquia que existe entre os objetos de um jogo.	25
Tabela 7 – Tarefas que foram utilizadas para analisar a nova versão da ferramenta.	33
Tabela 8 – Tabela com os resultados dos tempos da ferramenta antiga para as tarefas relacionadas a objetos do jogo: T3, T1.1, T1.2 e T1.3. . . . .	34
Tabela 9 – Tempo que leva para executar as tarefas relacionadas a criação de um botão nas duas versões da ferramenta. . . . .	40
Tabela 10 – Operações e seus respectivos tempos para o sistema KLM. . . . .	43
Tabela 11 – Resultado de análise KLM para ambas as versões da ferramenta. .	44

# Sumário

<b>1</b>	<b>Introdução</b>	<b>8</b>
1.1	O projeto CGT	8
1.1.1	Vínculo com o CNPQ	8
1.1.2	Trabalhos anteriores	9
1.1.3	Os módulos desenvolvidos para o projeto	9
1.2	Introdução ao problema	10
1.3	Disposição do documento	12
<b>2</b>	<b>Problema</b>	<b>13</b>
2.1	Objetos de um jogo	13
2.2	Dificuldades encontradas	14
2.2.1	Organização dos objetos	15
2.2.2	Painéis de configuração dos objetos	15
2.2.3	Pré visualização dos objetos	17
<b>3</b>	<b>Descrição das melhorias</b>	<b>21</b>
3.1	Resumo dos pontos de melhoria	21
3.2	A nova tela inicial da ferramenta	22
3.3	Descrição técnica	23
3.4	Organização dos objetos	24
3.4.1	Descrição da solução	24
3.4.2	Implementação técnica	24
3.5	Painéis de configuração	26
3.5.1	Descrição da solução	26
3.5.2	Implementação técnica	26
3.6	Pré visualização	29
3.6.1	Descrição da solução	29
3.6.2	Implementação técnica	30
<b>4</b>	<b>Estudo de caso</b>	<b>32</b>
4.1	Funcionalidades analisadas	32
4.2	Estimando as funcionalidades	34

4.2.1	Configurar um objeto (T1)	34
4.2.2	Selecionar um objeto (T3)	35
4.2.3	Configurar as dimensões de um objeto (T1.1)	35
4.2.4	Configurar a posição de um objeto (T1.2)	36
4.2.5	Configurar uma animação de um objeto (T1.3)	37
4.2.6	Configurar uma tela (T2)	39
4.2.7	Configurar a textura da tela (T2.1)	39
4.2.8	Criar um botão (T2.2)	40
4.2.9	Configurar textura de um botão (T2.2.1)	40
4.2.10	Posicionar um botão (T2.2.2)	41
4.2.11	Configurar o tamanho de um botão (T2.3)	42
4.3	Análise dos resultados	44
<b>5</b>	<b>Conclusão e trabalhos futuros</b>	<b>45</b>
	<b>Referências</b>	<b>47</b>
	<b>Anexos</b>	<b>48</b>
	<b>ANEXO A Manual da ferramenta</b>	<b>49</b>
A.1	Criar tela	49
A.2	Criar mundo	50
A.2.1	Critérios de vitória	50
A.2.2	Critérios de derrota	50
A.2.3	Configurar mensagens	51
A.3	Configurar <i>spritesheet</i> dos objetos	51
A.4	Ator principal	52
A.5	Configurar opositores	53
A.6	Configurar inimigos	54
A.7	Bônus	54
A.8	Configurar projétil	54
A.9	Configurar HUD	54
	<b>ANEXO B Diagramas de classes</b>	<b>68</b>



# 1 Introdução

Neste trabalho, é apresentado problemas existentes e pontos de melhorias na primeira versão da ferramenta de construção de jogos do projeto CGT (ferramenta 1.0). Bem como, de que modo essas questões foram tratadas e implementadas na segunda versão (ferramenta 2.0) que tem como objetivo melhorar o processo de criação de jogos e tornando-o mais intuitivo para o usuário final.

## 1.1 O projeto CGT

O projeto CGT que surgiu para tornar possível a qualquer pessoa a criação de jogos, assim como é explicado no trabalho (AQUINO, 2015), vem sido desenvolvido desde 2013 e possibilita a construção de jogos eletrônicos por qualquer pessoa, não sendo necessário conhecimentos em programação ou nas técnicas de criação de jogos. De forma fácil, simples e prática, a ferramenta, atende a necessidade da maior parte dos usuários, podendo produzir jogos de vários temas e para diversos fins, desde entretenimento a educação, por exemplo. Assim como, é proposto no *website* do projeto que explica, detalhadamente, o seu objetivo.

O projeto Ceará Game Tools tem como objetivo oferecer uma ferramenta para a construção de jogos. Em suma, qualquer um poderá criar seu próprio jogo utilizando componentes *drag and drop* e os configurando. (CGT, 2015a)

### 1.1.1 Vínculo com o CNPQ

A idealização e o início do desenvolvimento da ferramenta CGT deu-se no ano de 2013, através do projeto Ceará Game Tools (CGT) – Pesquisa, Desenvolvimento e Comercialização de Games Temáticos da Cultura Cearense, que é uma iniciativa do professor do IFCE Carlos Hairon Ribeiro Gonçalves apoiado pelo CNPQ<sup>1</sup> por meio do edital 80 de 2013 de número de processo 409227/2013-7.

A proposta deste projeto era desenvolver uma ferramenta de software livre para o desenvolvimento automático de *games* casuais em que qualquer pessoa com conhecimentos medianos de operação de microcomputadores poderia desenvolver jogos.

---

<sup>1</sup> Centro Nacional de desenvolvimento científico e tecnológico.

Neste caso, não há 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. Além disso, ela pode gerar jogos que podem ser executados em vários sistemas operacionais (*Windows*, *Linux*, *MacOS* e *Android*). Desse modo, os desenvolvedores podem aferir retorno financeiro com a venda e/ou popularização dos jogos, democratizando este canal para todas as pessoas que queiram ingressar nessa área.

### 1.1.2 Trabalhos anteriores

A partir do projeto vinculado ao CNPQ, a primeira versão da ferramenta foi produzida, trabalhando o desenvolvimento dos jogos apenas com entrada e manipulação de dados, utilizando apenas botões e comandos, sem componentes *drag and drop* ou módulo de pré-visualização. Com base no que foi desenvolvido ao longo desta etapa, foi produzido um trabalho acadêmico de conclusão de curso por título “Ceará Game Tools: Uma ferramenta de software livre para geração automática de games” (AQUINO, 2015).

Em aspectos gerais, esse trabalho apresenta todo o processo de criação da ferramenta Ceará Game Tools (CGT), que possui um ambiente de desenvolvimento visual simples e intuitivo e uma biblioteca de imagens e sons disponibilizados para os usuários desfrutarem. Além disso, nesse trabalho, relata trabalhos e *softwares* similares ao CGT, descreve os métodos e ferramentas utilizados para o desenvolvimento da aplicação, além da forma com ela foi desenvolvida, os módulos em que o CGT está dividido, os pacotes e classes que compõem a ferramenta, seus diagramas, benefícios, bem como trabalhos futuros e possíveis melhorias para a ferramenta. Nesta última parte, o autor destaca a importância da criação de novas políticas que aumentem as opções dos jogos e implementem novas estruturas, do aumento no número de plataformas de exportação dos jogos desenvolvidos, além de adicionar módulo de pré-visualização do jogo, o qual se concentra o assunto do presente trabalho.

### 1.1.3 Os módulos desenvolvidos para o projeto

O projeto CGT é composto por módulos que foram desenvolvidos e dividem entre si as funcionalidades existentes, eles são mostrados e descritos na tabela 1, o

módulo da ferramenta é o objeto de estudo para este trabalho e tem como objetivo implementar toda a interação com o usuário, além de ser responsável por apresentar a ele o jogo no momento da sua criação de forma intuitiva, simples e prática. Pode-se notar que, esse módulo é essencial para o projeto e deve ser capaz de lidar com tudo que possibilita a criação de jogos.

Módulos	Descrição
core	Contém os objetos essenciais para o projeto. Assim como: Ator, Inimigo, Bônus. Na secção 2.1 pode ser visto todos os objetos.
desktop	Responsável por possibilitar a execução do jogo no computador, em ambientes que rodam os sistemas <i>Windows</i> , <i>Linux</i> e <i>MacOS</i> .
ferramenta	Corresponde ao módulo objeto de estudo deste trabalho que possibilita a criação dos jogos.
android	Permite exportar o jogo para execução nos dispositivos que rodam o sistema operacional <i>Android</i> .
ios	Permite exportar o jogo para execução nos dispositivos que rodam o sistema operacional <i>iOS</i> .

Tabela 1 – Módulos existentes no projeto CGT.

## 1.2 Introdução ao problema

A identificação dos problemas e pontos de melhoria para a ferramenta foram percebidos, principalmente, através do uso e estão relacionados a interação com o usuário final, além disso, na forma como o jogo que está sendo produzido pelo usuário é percebido por ele mesmo. Para tanto, a ferramenta 1.0 possui uma interface bastante robusta e atende o propósito de criar os objetos que compõem o jogo e, eventualmente, produzi-lo em forma de aplicação para a plataforma destino. Entretanto, a mesma carece em fornecer algum *feedback* no momento da criação, obrigando ao usuário a sempre executar o jogo para poder perceber atributos essenciais dos objetos dele, tais como: a posição, o tamanho, a textura. Por exemplo, sejam as posições de um novo objeto configuradas da forma que se deseja, logo após isso, naturalmente, o usuário executa o jogo para poder visualizar o que configurou, e também entender o que representa graficamente o número que inseriu. Pode-se ver esse processo

ocorrendo na imagem 1, as duas janelas que estão abertas são, respectivamente, a do jogo em execução e da ferramenta de criação. Logo, podemos assumir que para todo objeto recém configurado, será preciso executar o jogo para verificar o resultado da configuração e com isso o processo de criação se torna bastante repetitivo.

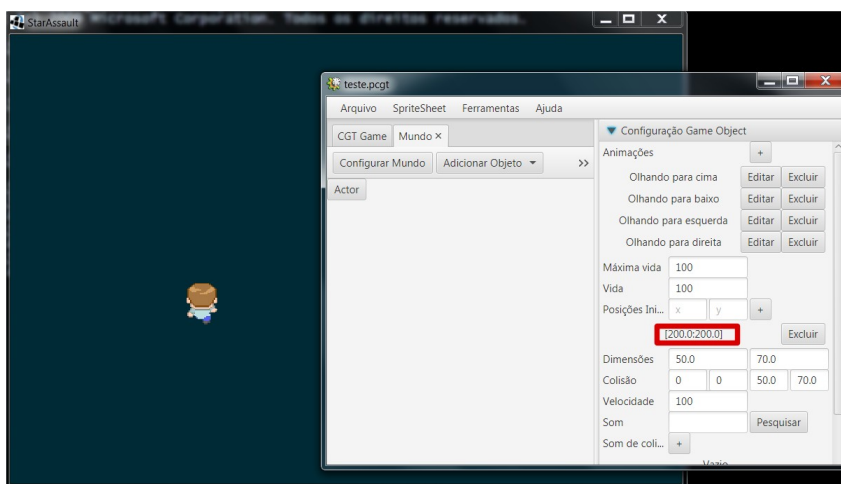


Figura 1 – Problema de pré-visualização na ferramenta 1.0.

Além disso, os controles que fazem parte da primeira versão da ferramenta, tornam, na maioria das vezes, oneroso o processo de configuração dos objetos. Podendo citar a configuração das animações de um objeto, é necessário criar as animações olhando no *spritesheet*<sup>2</sup> do objeto e preenchendo campos de texto com o valor da linha e coluna do *sprite* correspondente a animação configurada (ver imagem 2).

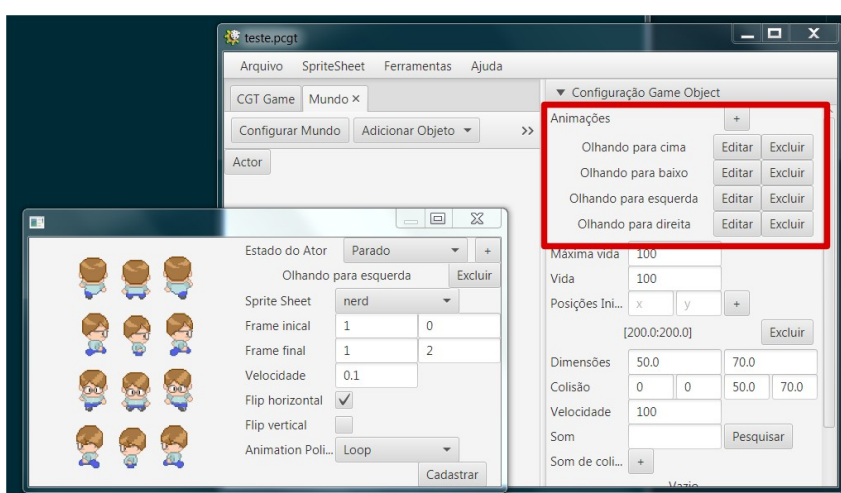


Figura 2 – Problema com os controles na configuração de uma animação na ferramenta 1.0.

<sup>2</sup> Imagem que contém todas as faces de um objeto no jogo.

Os problemas encontrados na ferramenta 1.0 foram a motivação para esse trabalho e serão descritos detalhadamente no capítulo 2.

## 1.3 Disposição do documento

Este documento segue o método que consiste em: apresentar um problema, propor sua solução e, por fim, estudar as consequências da solução adotada. Dessa forma, tudo estará dividido basicamente nas três partes seguintes:

**Definição do problema** Mostrado no capítulo 2 que tem como objetivo enumerar detalhadamente os problemas e pontos de melhoria encontrados na ferramenta 1.0 e como foram encontrados. Consequentemente, prepara-se uma lista das principais funcionalidades que farão parte da nova versão da ferramenta.

**Proposta de melhorias** Corresponde ao capítulo 3 que visa propor soluções para as questões encontradas anteriormente. Além disso, explicá-las e justificá-las detalhadamente, por fim demonstrando a solução adotada e como elas foram alcançadas.

**Análise dos resultados** É um estudo de caso, mostrado no capítulo 3.6.2, com a nova versão da ferramenta, onde busca-se comparar o tempo de criação de jogo, a facilidade de uso e a quantidade de esforço necessário para configurar.

## 2 Problema

A ferramenta de construção de jogos do projeto CGT é robusta e atende aos objetivos propostos. Contudo, ela possui lacunas quando se trata da interação humano computador (IHC). A falta de pré-visualização dos objetos contidos no jogo dificulta a percepção do usuário, obrigando-o a sempre executar o jogo para observar as configurações que foram feitas. Além disso, os controles não são claros e, assim, produzir um jogo pode acabar sendo uma experiência onerosa, confusa e repleta de configurações repetitivas. A nova versão da ferramenta visa resolver os dois problemas mencionados, principalmente o primeiro item, por ser mais importante para a usabilidade e clareza dos controles envolvidos no processo de criação de um jogo. Com isso em mente, faz-se necessário conhecer os objetos e controles que existem em um jogo da primeira versão da ferramenta, com o objetivo de perceber os problemas e as possíveis melhorias.

### 2.1 Objetos de um jogo

Para enumerar todos os problemas e melhorias da ferramenta 1.0, é importante construir um jogo com ela, então é preciso conhecer os objetos que existem nele e como eles são configurados. Observar que, cada um deles possui suas características e configurações, assim como um propósito no jogo, a tabela 2 mostra os possíveis tipos objetos e a descrição deles. Notar que, eles são inseridos na ferramenta a medida que o usuário ver necessidade e são responsáveis por dar liberdade ao autor de construir vários formatos de jogos, por exemplo jogos de: perguntas e respostas, concluir percurso, derrotar inimigo, sobreviver determinado tempo.

Objeto	Descrição
Mundo	Responsável por agregar todos os personagens de uma fase do jogo definindo o plano de fundo.
Ator	Objeto controlado pelo jogador.
Inimigo	Objeto que causa dano ao ator e impede os objetivos dele.
Opositor	Objeto que impede ações do ator, que não pode ser ultrapassado e pode ser destruído pelo ator.
Bônus	Objeto que promove bônus ao ator.
Projétil	Objeto que pode ser arremessado pelo ator.
Tela	Representa uma tela do jogo.
Botão de tela	Botão de uma tela do jogo.
Vida de um objeto	Representado através de uma barra, mostra a quantidade de vida que um objeto possui.
Munição de um objeto	A quantidade de projéteis que o ator ainda pode arremessar.

Tabela 2 – Objetos que existem em um jogo.

## 2.2 Dificuldades encontradas

A medida que a ferramenta foi utilizada, achou-se dificuldades, por exemplo, alguma interação confusa com a ferramenta que dificulta a criação e configuração de algum objeto, assim como o entendimento do papel dele no jogo, ou, até mesmo, a necessidade de executar o jogo para observar o que foi feito. Essas dificuldades, embora pareçam subjetivas, podem ser mensuradas e quantificadas normalmente. As demais seções desde capítulo, encarregam-se de trazer esses problemas, dentre eles existe, por exemplo, a dificuldade de observar um objeto, após alterar alguma propriedade dele. Notar que, a ferramenta 1.0 está disponível no [site do projeto CGT](http://www.cgt.ifce.edu.br/downloads.php) na seção de *downloads*<sup>1</sup>.

<sup>1</sup> <http://www.cgt.ifce.edu.br/downloads.php>

### 2.2.1 Organização dos objetos

Um dos pontos mais importantes quando se trata de criação de jogos, é que - no decorrer do desenvolvimento - eles podem se tornar enormes e, para a ferramenta, um jogo é um conjunto de objetos e suas respectivas propriedades. Logo, é imprescindível que os objetos possam ser vistos de forma lógica e, mais importante, organizada, já que, assim, serão acessados facilmente, pois o usuário terá necessidade de retornar a eles para melhorar ou corrigir alguma configuração. Além disso, vale ressaltar que entre os objetos existe uma relação de pertinência, um ator está inserido em um mundo, assim como o inimigo que ele precisa derrotar também o está, essa hierarquia precisa ser levada em consideração na ferramenta e o jogo será criado respeitando essas relações.

Na figura 3, pode-se visualizar um jogo sendo criado e - a partir dessa imagem - pode-se notar que a ferramenta é dividida em cinco áreas importantes (descritas na tabela 3). Os botões que representam objetos são organizados por colunas que representam as categorias: ator, inimigos, opositores, bonificações e informações da tela. A medida que é adicionado um novo objeto para a aba, este é posicionado em uma dessas colunas. Essa forma de organizar os objetos do jogo, possui a vantagem de ser objetiva, pois exibe apenas o conteúdo que importa para aquele agregador (uma fase do jogo ou uma tela). No entanto, a medida que o usuário vai inserindo elementos ao jogo e configurando eles, essa visão pode perder o significado, isto é, precisa-se de algo além de um botão com um nome gerado automaticamente para representar um objeto e ser identificado facilmente pelo usuário. Como pode ser visto na figura 3, nesse jogo, existem quatro inimigos configurados, mas não se sabe quais são as características deles apenas olhando para os botões, ou seja, o usuário precisa selecionar um a um até encontrar qual deles é o que ele procura, qual deles é o inimigo final da fase, por exemplo. Sendo assim, a ação de reconfigurar um objeto pode tornar-se difícil a medida que novos itens forem sendo adicionados ao jogo.

### 2.2.2 Painéis de configuração dos objetos

Todos os itens que fazem parte de um jogo possuem características que os definem, sendo assim cabe a ferramenta prover ao usuário final uma forma clara e objetiva de configurá-las. Além disso, a ferramenta tem obrigação de validar os



<b>Menu</b>	No menu é possível realizar ações que refletem no projeto todo, por exemplo: salvar, adicionar <i>spritesheet</i> , executar o jogo.
<b>Abas</b>	As abas representam os agregadores de objetos, podem conter a aba inicial do projeto, um <i>mundo</i> ou uma tela.
<b>Toolbar</b>	Permite realizar ações referentes ao agregador, se for de um mundo possibilitar adicionar um inimigo ao jogo, por exemplo.
<b>Botões de objetos</b>	Cada objeto é representado por um botão que pertencem a uma determinada aba.
<b>Painel lateral</b>	Mostra as configurações do objeto selecionado, ao clicar em qualquer botão será mostrado as configurações daquele objeto.

Tabela 3 – Controles utilizados na primeira versão da ferramenta.

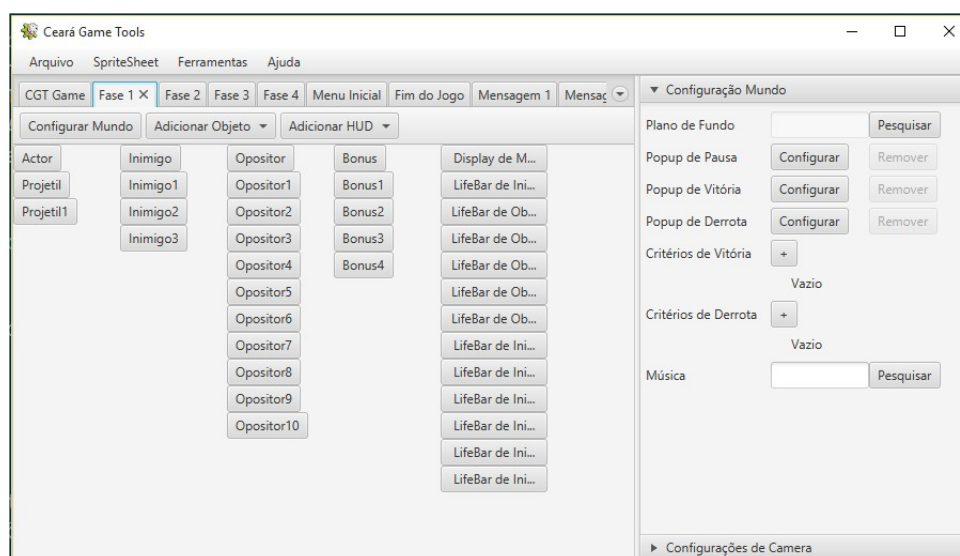


Figura 3 – Disposição dos objetos na versão anterior da ferramenta.

dados e guiar o usuário nos casos mais complexos, assim como dar um significado para o dado que está sendo recebido. Por exemplo, vamos supor que exista um objeto recém adicionado no jogo e que pretendemos configurar a sua posição inicial na tela, a ferramenta recebe essa configuração através de dois números que são, respectivamente, as posições cartesianas  $x$  e  $y$  do objeto, deve-se verificar se essas posições estão contidas no retângulo que representa a tela do jogo, assim como fornecer ao usuário formas de saber o que significa o valor que ele está configurando. Essa e outras propriedades possuem a mesma necessidade de serem visualizadas,

ver tabela 4.

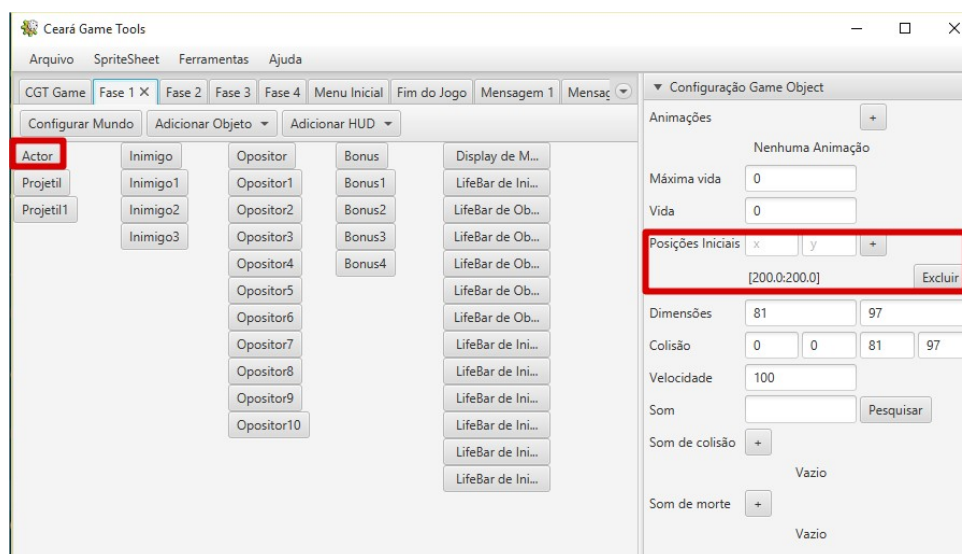


Figura 4 – Configuração da posição inicial de um objeto na primeira versão da ferramenta.

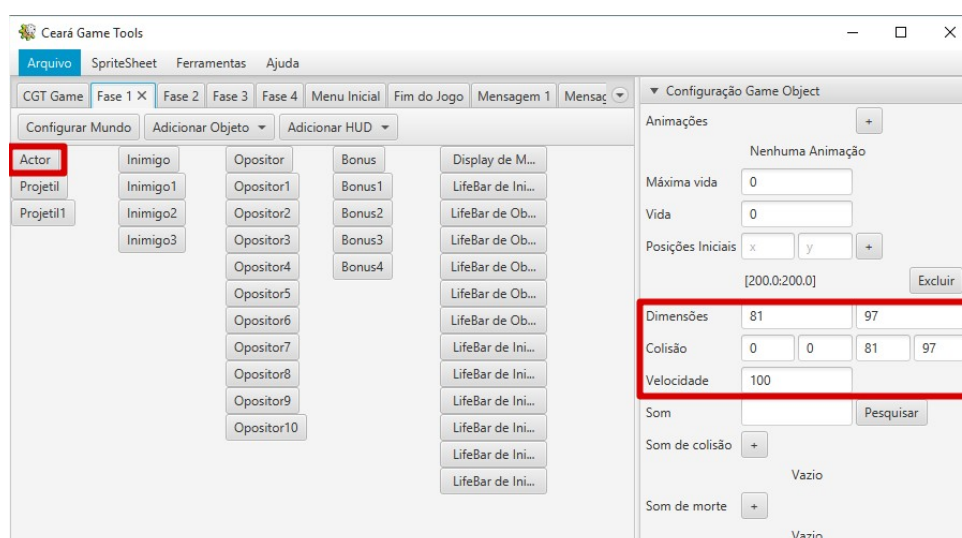


Figura 5 – Configuração das dimensões de um objeto na primeira versão da ferramenta.

### 2.2.3 Pré visualização dos objetos

Os principais problemas dos painéis de configuração da ferramenta 1.0 estão diretamente relacionados a falta de pré visualização dos objetos, notar que não há forma melhor de perceber o significado de uma configuração do que visualizando-a. Ou seja, simular o jogo é importantíssimo para a criação. Logo, a maior motivação para a criação do módulo que é proposto neste trabalho é a falta desse recurso.

A visualização dos objetos também contribui para os problemas mencionados nas secções anteriores. A disposição deles principalmente, pois com a prévia dos objetos, torna-se fácil identificá-los. Na secção 3 será mostrado a solução sugerida para cada um desses problemas. Mas, antes disso, é importante enumerar as propriedades dos objetos que se beneficiariam com a pré-visualização, ou seja, as propriedades que trazem ao usuário a necessidade de executar o jogo para aferir a sua respectiva configuração. Esses atributos são mostrados na tabela 4.

Objeto(s)	Atributo(s)
Mundo e tela do jogo	Plano de fundo (figura 6).
Ator, inimigo, bônus, opositor e projétil	Animações ( <i>spritesheet</i> ), posição inicial, dimensões e área de colisão (figura 7).
Botão de uma tela	Posição, dimensões, textura do botão normal e textura quando for pressionado (figura 8).
Munição do projétil	Posição, dimensões e ícone (figura 9).
Barra de vida de um objeto	Posição, dimensões, textura do preenchimento da barra e textura do plano de fundo (figura 10).

Tabela 4 – Objetos do jogo e suas respectivas propriedades que demandam ser simuladas, ou seja, serem visualizadas na criação.

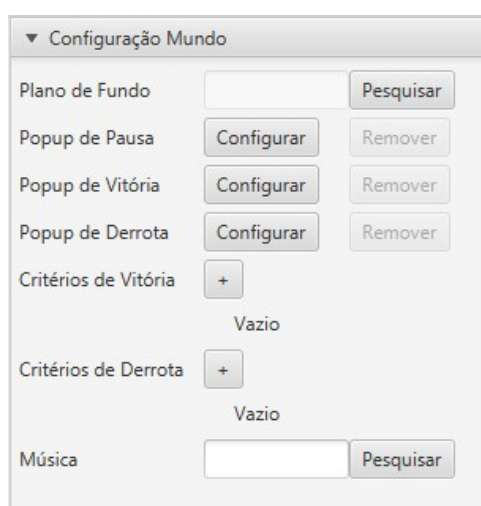


Figura 6 – Painel de configuração de um mundo do jogo na primeira versão da ferramenta.

**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

Figura 7 – Painel de configuração de um objeto do jogo, existe para os seguintes objetos: ator, inimigo, bônus, opositor e projétil.

**Configurações Button Screen**

Ir para ▼

Textura  Procurar

Textura Press.  Procurar

Relativo X

Relativo Y

Largura relativa

Altura relativa

Figura 8 – Painel de configuração de um botão na tela do jogo.

**Configurações Display de Munição**

Relativo X

Relativo Y

Largura relativa

Altura relativa

Icon Selecionar

Projectile ▼

Figura 9 – Painel de configuração do mostrador de munição de um projétil do jogo.

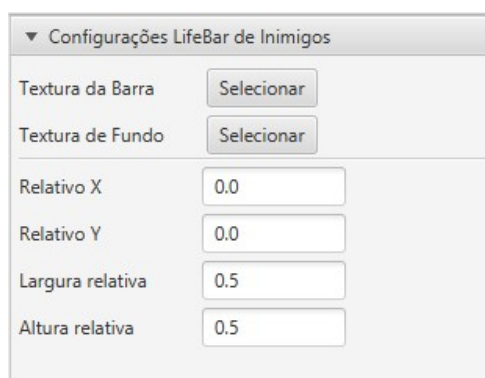


Figura 10 – Painel de configuração da barra de vida de um objeto do jogo.

## 3 Descrição das melhorias

Tendo visto os problemas da primeira versão da ferramenta mostrados na secção 2, discuti-se aqui como resolvê-los e o que é necessário para isso. Primeiramente, tem-se uma solução correspondente aos problemas indicados, daí é discutido a sua viabilidade e a implementação técnica. Vale salientar que, os pontos de melhorias mostrados encontram-se implementados na ferramenta 2.0, dessa forma pode ser mostrado o resultado disso (secção 3.6.2).

### 3.1 Resumo dos pontos de melhoria

Para cada um dos problemas indicados anteriormente, é necessário determinar a solução de cada um e, além disso, detalhar qual foi o meio técnico. Portanto, tem-se três problemas descritos nas secções 2.2.1, 2.2.2 e 2.2.3 e, conseqüentemente, as respectivas melhorias que estão relacionadas a seguir.

- a) **Organização dos objetos:** Utilizou-se uma árvore que organiza todos os objetos do jogo e deixa claro a relação que existe entre cada um deles. A árvore de objetos possui como raiz o projeto e cada objeto inserido nele é mostrado como seu filho. Com isso, os botões que representam os itens do jogo na primeira versão da ferramenta foram substituídos por essa árvore.
- b) **Painéis de configuração dos objetos:** Os painéis de configurações permanecem parecidos com a primeira versão da ferramenta, as melhorias visam permitir que as informações alteradas reflitam sempre com a área destinada a simular o jogo, a área de pré visualização, ou seja, no módulo de pré visualização, os painéis terão que interagir para mostrar as mudanças feitas no objeto.
- c) **Pré visualização dos objetos:** A melhoria mais importante desse módulo, consiste em mostrar uma prévia do jogo ao usuário, mostrando os objetos inseridos e configurados, permitindo que o usuário perceba melhor o jogo que ele está criando.

## 3.2 A nova tela inicial da ferramenta

Com o objetivo de implementar as melhorias foi necessário redesenhar a tela inicial da ferramenta, colocando os componentes novos e retirando os que não são mais utilizados. Dessa forma, a árvore de objetos do jogo está localizada a direita da tela, os painéis de configuração dos objetos ficam logo abaixo dela. E, a área de pré visualização é disponibilizada no centro da ferramenta. A figura 11 mostra a nova versão da tela inicial e a tabela 5 descreve os controles utilizados.

<b>Menu</b>	Assim como na primeira versão, armazena comandos da ferramenta, a partir do menu pode-se adicionar objetos no jogo, na nova versão da ferramenta o menu substitui a <i>toolbar</i> e possui atalhos para facilitar o uso. Por exemplo, CTRL + I para adicionar um inimigo ao jogo.
<b>Área de pré visualização</b>	Região destinada a prévia do jogo, onde os objetos são exibidos da forma mais próxima possível ao jogo. O usuário pode interagir com essa área, selecionando um objeto ou posicionando ele em outro lugar da tela por exemplo.
<b>Árvore de objetos</b>	Componente que sumariza todos os objetos existentes no jogo. A árvore é responsável por exibir os objetos em sua hierarquia.
<b>Painel de configuração</b>	Painel de configuração do objeto que foi selecionado. Nessa região todas as configurações relacionadas ao objeto selecionado devem ser exibidos

Tabela 5 – Controles utilizados na segunda versão da ferramenta.

Os novos controles interagem entre si com o seguinte comportamento, a cada interação na árvore a ferramenta atualiza os painéis e a área de pré visualização de acordo com o objeto que esteja selecionado. Este que, por sua vez, é atualizado também quando algumas propriedades são alteradas no painel. A tabela 4 mostra os atributos que quando alterados atualizam a prévia, por exemplo, o campo correspondente ao plano de fundo do mundo (imagem 11). A seguir, veremos como cada um dos problemas enumerados anteriormente foram resolvidos com base na solução sugerida, além disso como foi a implementação.

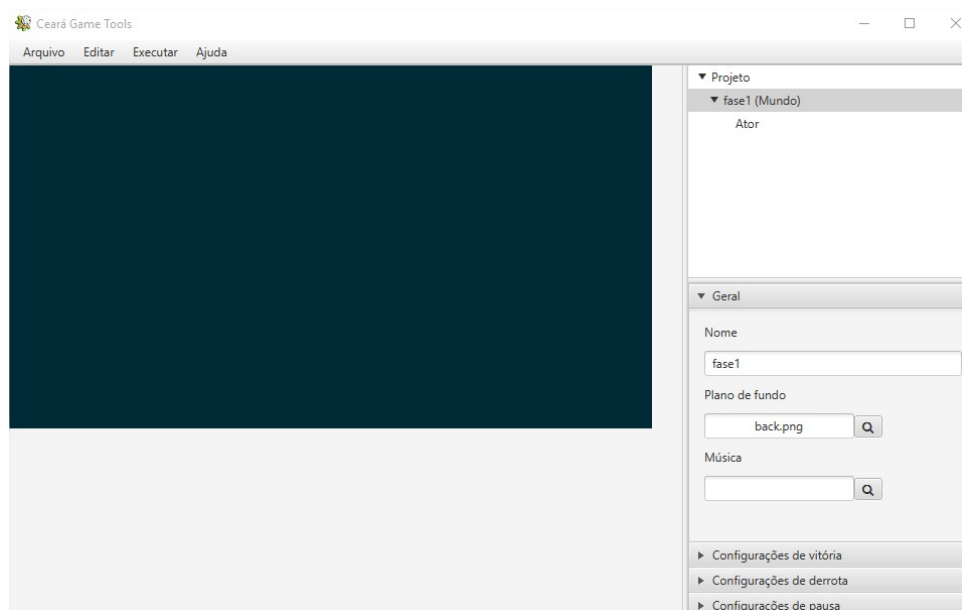


Figura 11 – Nova tela inicial da ferramenta com os novos controles que possibilitaram as melhorias.

### 3.3 Descrição técnica

A ferramenta é escrita na linguagem Java e utiliza o *framework* JavaFX para a interface do usuário, todo o código escrito para a primeira versão da ferramenta foi reutilizado, incrementando-o a medida que foi preciso na implementação do módulo de *preview*.

Em *JavaFX*, é usado um arquivo XML que descreve o *layout* de uma tela e uma classe que cuida do comportamento dela. Assim, a implementação do módulo de pré visualização, consistiu em alterar esse arquivos XML e suas respectivas classes e, além disso, classes do pacote `br.ifce.edu.cgt.application.vo`, que receberam objetos customizados que implementam a interface `DrawableObject`. A interface `DrawableObject` é usada por todos os objetos que são mostrados na área de pré visualização e árvore de objetos, pois define o comportamento que deverá ter os objetos que estarão contidos nessas regiões. Nessa interface, existem dois métodos que são responsáveis por desenhar o objeto e o seu painel de configuração, métodos `drawObject()` e `drawConfigurationPanel()` respectivamente.

Com isso é possível visualizar as implementações que foram necessárias para concretizar o objetivo do módulo de pré visualização. O diagrama de classes, mostrando todas as classes que representam esse módulo, pode ser visto nos anexos.



## 3.4 Organização dos objetos

### 3.4.1 Descrição da solução

Árvores são ideais para organizar elementos que estão classificados hierarquicamente, com isso, tornou-se a melhor maneira de organizar os objetos da nova versão da ferramenta CGT. A versão anterior exibia os mundos criados como abas e, dentro de cada uma, os objetos criados eram botões que quando eram clicados exibiam o painel de configuração correspondente ao objeto que foi clicado. Ao substituir essa visão pela atual, permite-se que, facilmente, o usuário tenha visão de tudo que existe no jogo sem a necessidade de alternar entre abas, vale lembrar que a árvore de objetos ocupa menos espaço na janela, o que permitiu colocar outros controles. Então, essa melhoria é mais lógica e conveniente, pois agrega mais praticidade a ferramenta. A imagem 12 mostra um exemplo de uma árvore de objetos para um jogo em criação.



Figura 12 – Árvore de objetos da nova versão da ferramenta.

### 3.4.2 Implementação técnica

O objetivo principal da árvore de objetos é garantir que a ferramenta seja atualizada quando um item for selecionado, pois o mesmo deve ser exibido na área de pré visualização e nos painéis de configuração, mas também ela deve organizar os objetos, agrupando-os quando necessário.

Usa-se o objeto `TreeView` para representar a árvore, onde cada um dos itens é um `DrawableObject`. O comportamento que queremos que seja feito, foi implementado com uma fábrica de células customizada que é uma classe que estende de `TreeCell`

e implemente os métodos `startEdit` e `updateItem`, chamados quando o usuário interage com a árvore. Nesses métodos, coloca-se a instrução para que seja desenhado os objetos e o painel, ou seja, executa-se os métodos abstratos de `DrawableObject` (`drawObject` e `drawConfigurationPanel`), por fim, atribui-se ao `TreeView` a fábrica customizada assim como é mostrado abaixo (seja `t` uma instância de `TreeView`):

```
t.setCellFactory(new Callback<TreeView<DrawableObject>, TreeCell<DrawableObject>>(){
    @Override
    public TreeCell<DrawableObject> call(TreeView<DrawableObject> param) {
        return new DrawableObjectTreeCellImpl();
    }
});
```

Além do comportamento de redesenhar de acordo com o que foi selecionado na árvore, é importante definir os itens do jogo de acordo com a sua hierarquia, já que a árvore é montada obedecendo isso, a tabela 6 mostra todos os itens e os seus respectivos itens superiores.

Item	Item superior
Projeto	(Raiz)
Mundo	Projeto
Ator, Inimigos, Bônus(es), Opositor(es)	Mundo
Projétil	Ator
Barra de vida	Ator, Inimigo
Munição	Projétil
Tela	Projeto
Botão de tela	Tela

Tabela 6 – Tabela mostrando a hierarquia que existe entre os objetos de um jogo.

Garantir que a ferramenta mostre todos os objetos de acordo com a hierarquia, consiste em adicioná-los corretamente, o `PreviewPane` faz isso nos métodos que são chamados pelos eventos do menu editar, `addEnemy` por exemplo.

## 3.5 Painéis de configuração

Os painéis de configuração na ferramenta são, a grosso modo, os controles responsáveis por receber as informações do usuário e transformá-las em objetos do jogo. É importante que, os controles usados sejam claros, objetivos e consigam passar para o usuário o significado do que está sendo configurado.

### 3.5.1 Descrição da solução

Para a implementação do módulo de pré visualização foi necessário aprimorar os painéis existentes na primeira versão da ferramenta, possibilitando as funcionalidades da segunda versão, tem-se a necessidade, por exemplo, de não exibir diálogos de configuração com tanta frequência, como é feito na versão anterior, pois os mesmos se posicionam na frente da ferramenta e, conseqüentemente, da área de pré visualização (ver figura 2 como exemplo de configuração em dialogo). Além disso, é necessário para os objetos que possuem animação uma prévia dela ou uma forma de selecioná-la melhor no painel de configuração, é o caso do ator e inimigo por exemplo. Lembrar que, o objetivo é passar para o usuário o que uma determinada configuração representa.

Os painéis são mostrados na ferramenta de forma semelhante aos objetos na pré visualização, ou seja, são disparados através da árvore de objetos, a medida que o usuário seleciona um objeto dela, a configuração correspondente é exibida na região destinada aos painéis. Em um painel de configuração de um objeto, pode possuir mais de uma configuração para ele, cada configuração deve ser referente a algo e elas são exibidas separadamente. Usa-se o componente de acordeão para agregar as configurações no painel de acordo com o objetivo dela. Ver a figura 13 por exemplo, nela temos um painel de configuração de um ator do jogo, percebe-se que cada configuração é separada de acordo com sua finalidade, há uma aba para configurar as animações e outra para configurar as propriedades relacionadas a colisão, por exemplo.

### 3.5.2 Implementação técnica

A implementação dessa melhoria está relacionada a reutilizar os painéis que já existiam e melhorar alguns aspectos, dentre eles, pode-se destacar a configuração

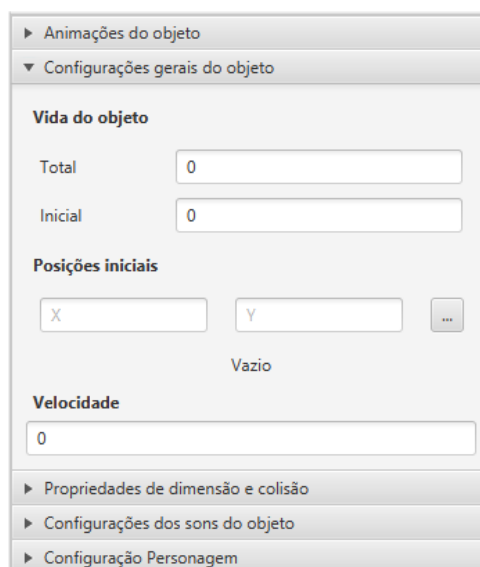


Figura 13 – Painel de configuração de um ator no jogo

das animações do painel responsável por configurar um *game object*. Na primeira versão da ferramenta, uma animação era adicionada a partir de uma janela *pop up* que mostrava o *sprite* do objeto, então o usuário escolhia dentro de uma matriz o *frame* inicial e final, assim como a finalidade da animação e outras propriedades dela (ver imagem 14). Quando há um painel destinado a mostrar as configurações do objeto

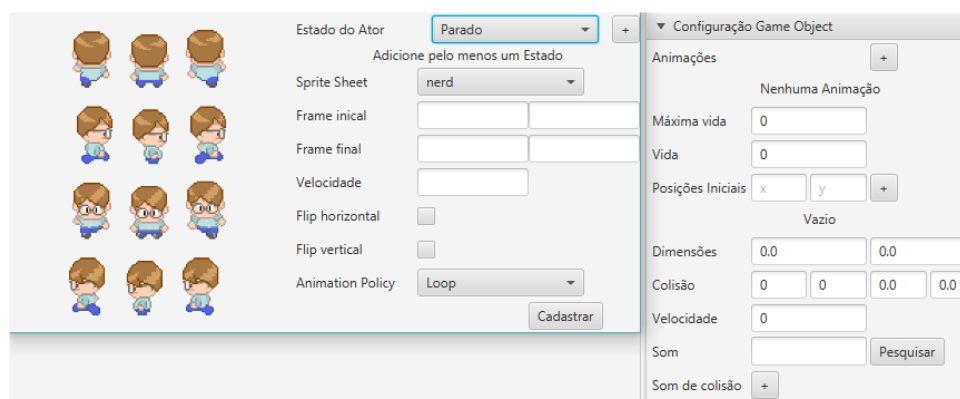


Figura 14 – Janela para adicionar animações para um objeto na primeira versão da ferramenta.

correspondente, faz-se desnecessário o uso de janela *pop up*, dessa forma, na nova versão da ferramenta as configurações da animação do objeto foram colocadas no painel dele apenas, facilitando a pré visualização e também o manuseio. Ver imagem 15, notar que, o que é preciso para configurar uma animação está em três abas desse painel, onde a última aba é uma lista de todas as animações que já existem, a segunda as propriedades da animação e a primeira aba a configuração do quadro inicial e final

da animação. Vale notar que o comportamento de selecionar o quadro inicial e final de uma animação para um objeto, tornou-se mais fácil, pois antes era feito com campos de texto que recebiam as coordenadas  $x$  e  $y$  do quadro e, na segunda versão, faz-se necessário apenas clicar no quadro que se deseja. Logo, foi possível transferir tudo para o painel, contribuindo para a coesão da ferramenta e melhorando também no objetivo de pré visualização dos objetos.

De um ponto de vista técnico, os painéis implementados para a nova versão da ferramenta são classes que carregam o arquivo XML que descreve o *layout* (FXML) e, Consequentemente, definem o comportamento. Além disso, faz-se necessário escrever na função que abre um jogo salvo previamente o comportamento de preencher as informações desses painéis.

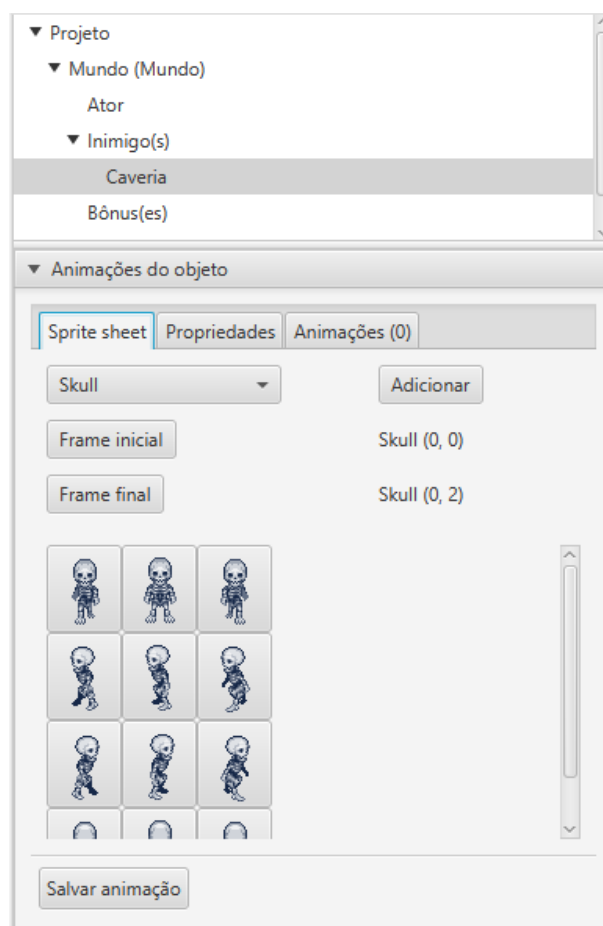


Figura 15 – Painel de configuração das animações do objeto na nova versão da ferramenta.

## 3.6 Pré visualização

A pré visualização dos objetos do jogos criados pela ferramenta CGT é a principal melhoria do módulo que foi proposto e desenvolvido com esse trabalho. Vale lembrar que, a nova versão da ferramenta consiste exatamente nos itens presentes neste capítulo, sendo assim, a pré visualização dos objetos é a última funcionalidade que foi implementada e a mais importante, pois como - será visto posteriormente - é a maior responsável por melhorar a experiência do usuário, comparando-a com a versão anterior.

### 3.6.1 Descrição da solução

Poder conferir o que está sendo produzido, é a principal falta que os usuários da primeira versão da ferramenta sentem. Dessa forma, pode-se dizer que é o maior problema que ela possui. Para resolvê-lo, foi preciso dedicar uma região da janela para uma área de pré visualização do jogo, com isso a ferramenta foi alterada para conter essa área, a árvore de objetos e o painel de configuração (figura 11). A área de pré visualização deve conter todos os objetos existentes no jogo para aquele agregador (mundo ou tela do jogo), além disso, o itens devem ser exibidos tais quais seriam no momento de execução do jogo, devem ser fiéis a configuração feita e refletir no resultado real. Na figura 16, pode-se ver como a tela da ferramenta é usualmente vista pelo usuário, notar que - na área de visualização - é mostrado um mundo em que podem ser vistos dois objetos configurados um ator e um inimigo.

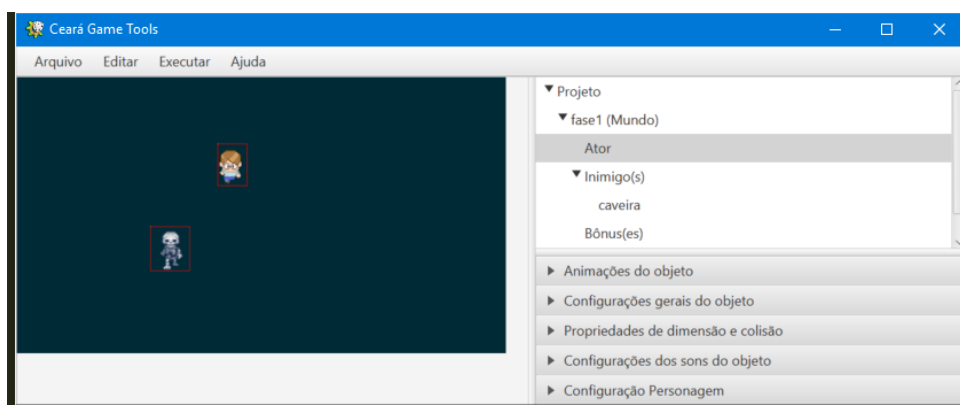


Figura 16 – Tela da ferramenta mostrando a área de pré visualização com dois objetos configurados.

### 3.6.2 Implementação técnica

A construção da pré visualização da ferramenta foi implementada utilizando objetos customizados que possuem três atributos essenciais para o funcionamento dessa funcionalidade:

- a) Um atributo que corresponde ao objeto raiz do jogo, por exemplo: ator, inimigo;
- b) Um método para desenhar esse objeto raiz na área de pré visualização;
- c) Um método para exibir o painel de configuração correspondente ao objeto selecionado;

Dessa forma, temos uma interface criada para esse fim, a `DrawableObject`, a partir daí, foi implementada uma classe abstrata, cujo nome é `AbstractDrawableObject`, responsável por implementar algumas das tarefas comuns de todos esses objetos desenháveis. Com isso, pode-se ainda estender essa classe abstrata e gerar classes específicas aos objetos que necessitam de uma prévia, então foram criadas classes com o sufixo `Drawable`, por exemplo `CGTGameActorDrawable` que tem o objetivo de descrever cada uma das propriedades citadas anteriormente.

Então, por definição, um objeto desenhável deve prover de meios para representar um objeto raiz do projeto, que são os objetos que estão no pacote `cgt.core` dentro do módulo `core` do projeto (ver tabela 1). Os meios que um desenhável possui são os métodos `drawObject` e `drawConfigurationPanel` que são responsáveis por exibir o objeto e o painel de configuração respectivamente. Esses objetos desenháveis, são criados pelo usuário e inseridos na árvore de objetos (assim como explicado na seção 2.2.1), no momento de inserção, o método que os desenha é chamado e, então, a pré visualização é exibida. A medida que, o usuário vai configurando esses objetos no respectivo painel de configuração, a prévia dele é atualizada na área de visualização. Isso é possível, graças a criação de métodos do tipo *callback* que existem em todos os painéis de configuração que necessitam que as mudanças sejam exibidas assim que feitas, notar que, esse método é chamado pelo próprio painel toda vez que algum atributo relevante é modificado, por exemplo, a dimensão e posição de um objeto, assim como é mostrado na figura 17.

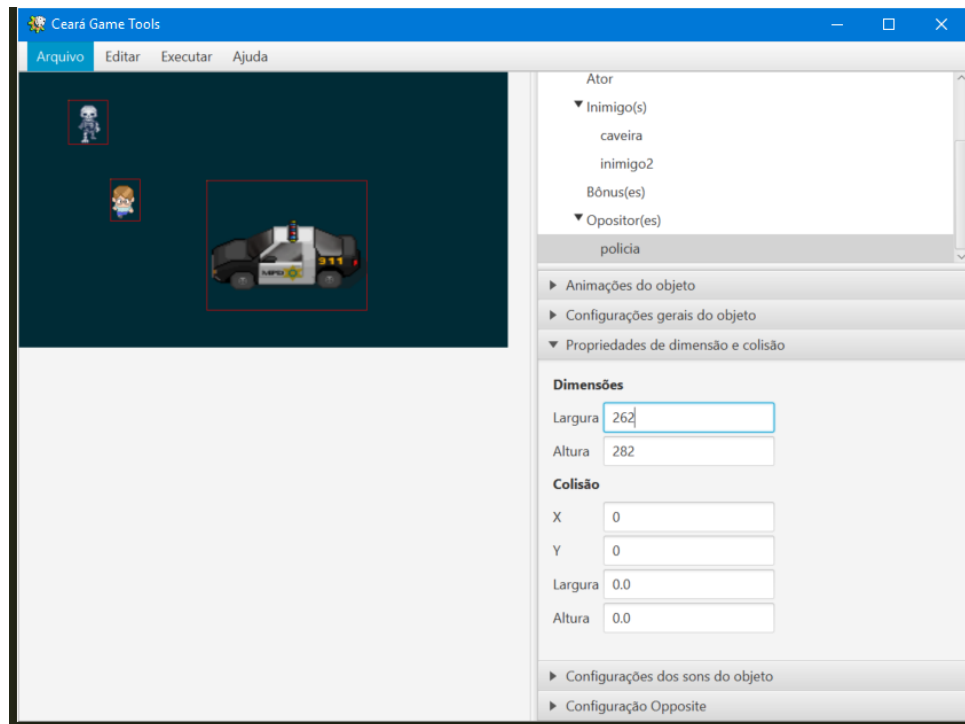


Figura 17 – Exemplo de dimensão do objeto sendo percebida no momento que é alterada na configuração

```

boundsH.focusedProperty().addListener(new ChangeListener<Boolean>() {
    @Override
    public void changed(ObservableValue<? extends Boolean> observable,
        Boolean oldValue, Boolean newValue) {
        if (!newValue) {
            /** comportamento para mudar a altura do objeto */
        }

        if (onUpdateRunnable != null) {
            onUpdateRunnable.run();
        }
    }
});

```

Ou seja, toda vez que é detectado um evento de mudança no campo de texto do tamanho do objeto, o método de *callback* - que é um objeto Java da classe `Runnable`, mas no painel é um atributo cujo nome é `onUpdateRunnable` - é executado da forma é mostrado a seguir. Notar que, quem cria o objeto que corresponde ao painel de configuração é que deve determinar o conteúdo desse método *callback*, dessa forma, é garantido que o painel seja compatível com a ferramenta.



## 4 Estudo de caso

Neste capítulo será realizado um estudo para mostrar o ganho que se obteve com a implementação do módulo de pré-visualização, ou seja, qual o verdadeiro resultado que foi obtido após utilizar a nova versão da ferramenta. Sendo assim, será feito comparações com as duas versões da ferramenta com o intuito de demonstrar avanço na criação de jogos. Ao colocar o módulo de pré visualização na ferramenta, sabe-se que, com isso, exclui-se a necessidade de frequentemente executar o jogo para verificar qualquer configuração feita. Então, na tabela 4 pode ser visto os atributos de cada objeto que fazem necessário a execução do jogo após a alteração deles. A nova versão da ferramenta elimina esse problema, ou seja, enquanto que para um jogo com cinquenta objetos criado na primeira versão da ferramenta, provavelmente, faria com que o usuário executasse esse jogo cinquenta ou mais vezes para averiguar o seu trabalho, a nova versão pouparia isso e exibiria todos os objetos com suas respectivas propriedades de uma só vez de forma objetiva e clara na área de pré visualização e na árvore dos objetos auxiliando no acesso a eles.

### 4.1 Funcionalidades analisadas

Para analisar o resultado deve-se criar um jogo com a antiga ferramenta e com a nova, pode ser visto no manual da ferramenta antiga, onde é mostrado como criar um jogo nela (AQUINO, 2015) e - com o anexo A deste documento - mostra-se como proceder com as alterações do módulo de pré visualização, ou seja como criar um jogo com a nova versão da ferramenta. Posto isso, será usado o modelo KLM (*Keystroke-Level Model*) que propõe quantificar o tempo de execução que o usuário leva para executar uma tarefa, será utilizado esse método na ferramenta para averiguar se as tarefas são executadas mais rápidas na nova versão.

Assim como sugere o modelo KLM (CARD; MORAN, 1980) há uma tarefa maior que a ferramenta deve possibilitar que é a tarefa de criar um jogo, mas ela pode ser fragmentada em tarefas menores que são listadas na tabela 7 e, além disso, é importante lembrar que elas podem ser executadas em mais de um cenário, o que para

o método KLM deve ser tratado de forma diferente (KIERAS, 2001). Notar que, essas tarefas assemelham-se bastante com os casos de uso da ferramenta, contudo será analisado apenas o que é relevante para averiguar as melhorias implementadas na nova versão da ferramenta.

ID	Tarefa	Descrição
T1	Configurar os objetos de um mundo:	Os objetos do mundo que entram nessa categoria podem ser: ator, inimigo, opositor, bônus e projétil.
T1.1	Configurar as dimensões;	Determinar e perceber o tamanho do objeto e o espaço de colisão.
T1.2	Configurar a posição;	Determinar e perceber alguma das posições iniciais do objeto.
T1.3	Configurar uma animação;	Para qualquer objeto criado, inserir apenas uma animação para um objeto.
T2	Configurar uma tela:	Criar uma tela para o jogo.
T2.1	Configurar textura;	Selecionar e perceber a textura para a tela.
T2.2	Criar botão:	Incluir um botão em uma tela criada.
T2.2.1	Configurar textura;	Configurar a textura do botão, inclusive quando ele for clicado.
T2.2.2	Posicionar botão;	Determinar a posição que o botão deve ficar na tela.
T2.2.3	Configurar tamanho;	Determinar o tamanho desse botão com relação a tela dele.
T3	Selecionar um objeto	Significa selecionar um objeto qualquer do jogo.

Tabela 7 – Tarefas que foram utilizadas para analisar a nova versão da ferramenta.

Acredita-se que as tarefas mostradas na tabela 7 já possam exemplificar o que se deseja, pois algumas tarefas muito importantes para a ferramenta não foram contabilizadas, tais como: selecionar um objeto, configurar um mundo, configurar especificamente um objeto do mundo (ator, inimigo, opositor, bônus e projétil).

## 4.2 Estimando as funcionalidades

Para estimar as tarefas, considere o sistema KLM e suas operações vistas na tabela 10. A partir dessas operações, pode-se expressar cada tarefa, uma vez que se sabe no que cada uma delas consiste. Nas próximas secções, é mostrado o que há em cada uma das tarefas listadas, começando com as atividades menores até as maiores.

### 4.2.1 Configurar um objeto (T1)

Assume-se que para criar um objeto, independente da versão da ferramenta, deve ser realizado os seguintes passos listados a seguir:

1. Selecionar um objeto (T3);
2. Configurar as dimensões (T1.1);
3. Configurar a posição (T1.2);
4. Configurar uma ou mais animações (T1.3);

Posto isso, considere a função  $tempo(x)$  que representa o tempo de uma dada tarefa  $x$ , então, pode-se dizer que tempo total para criar um objeto é

$$tempo(T1) = tempo(T3) + tempo(T1.1) + tempo(T1.2) + N \times tempo(T1.3)$$

onde  $N$  é a quantidade de animações que o objeto possui.

Tarefa	Tempo(s)	Tempo(s)
T3	5,0	2,5
T1.1	7,54	7,54
T1.2	2,76	2,4
T1.3	28,90	20,96

Tabela 8 – Tabela com os resultados dos tempos da ferramenta antiga para as tarefas relacionadas a objetos do jogo: T3, T1.1, T1.2 e T1.3.

Nas subsecções seguintes - secções 4.2.2, 4.2.3, 4.2.4 e 4.2.5 - é mostrado como foram calculados os tempos para cada uma das tarefas (ver tabela 8), com isso,

pode ser substituído os valores na equação anterior, assumindo que seja um objeto com sete animações, daí temos para a primeira versão da ferramenta o resultado de

$$\text{tempo}(T1_{\text{antiga}}) = 5 + 7,54 + 2,76 + 7 \times 28,98 = 15,3 + 202,86 \approx \mathbf{3m38s}$$

e para a ferramenta nova o resultado é

$$\text{tempo}(T1_{\text{nova}}) = 2,5 + 7,54 + 2,4 + 7 \times 20,96 = 12,44 + 146,72 \approx \mathbf{2m39s}.$$

Com isso temos o tempo que o usuário leva em média para configurar um objeto nas duas versões da ferramenta.

#### 4.2.2 Selecionar um objeto (T3)

Consiste em um objeto qualquer do jogo, seja ele contido em mundo ou um botão de tela, a ferramenta deve possibilitar que essa ação seja mais fácil possível, pois é uma das ações mais usadas pelo usuário. Com isso, considere os seguintes passos para selecionar um objeto na ferramenta a seguir.

##### Primeira versão:

1. Achar aba do objeto (M);
2. Posicionar ponteiro na aba (P);
3. Clicar na aba (BB);
4. Achar o botão do objeto (M);
5. Posicionar ponteiro no botão do objeto (P);
6. Clicar no botão do objeto (BB);

##### Nova versão:

1. Achar objeto na árvore (M);
2. Posicionar ponteiro no item (P);
3. Clicar no item do objeto (BB);

$$\begin{aligned} \text{tempo}(T3) &= 2\text{tempo}(M) + 2\text{tempo}(BB) + \text{tempo}(T3) = \text{tempo}(M) + \text{tempo}(BB) + \\ 2\text{tempo}(P) &= 2,4 + 0,4 + 2,2 = \mathbf{5,0s} \quad \text{tempo}(P) = 1,2 + 0,2 + 1,1 = \mathbf{2,5s} \end{aligned}$$

#### 4.2.3 Configurar as dimensões de um objeto (T1.1)

A tarefa de configurar as dimensões do objeto é, geralmente, a primeira a ser feita quando um novo objeto em configurado. Para essa tarefa, vale observar

que no caso da segunda versão da ferramenta, as dimensões já são preenchidas automaticamente com o valor calculado através do *spritesheet* fornecido. A seguir, os passos e operações para o pior caso:

**Primeira versão:**

1. Posicionar e clicar em *Configurações Game Object* (P+BB);
2. Posicionar e clicar no campo de texto *Dimensões* (P+BB);
3. Digitar novo valor da largura (múltiplos K, mínimo 4);
4. Repete os dois últimos passos 3 vezes para a altura, altura e largura do espaço de colisão.

**Nova versão:**

1. Posicionar e clicar em *Propriedades de dimensão e colisão* (P+BB);
2. Posicionar e clicar no campo de texto *Dimensões* (P+BB);
3. Digitar novo valor da largura (múltiplos K, mínimo 4);
4. Repete os dois últimos passos 3 vezes para a altura, altura e largura do espaço de colisão.

$$\begin{aligned} \text{tempo}(T1.1) &= 5\text{tempo}(BB) + 12\text{tempo}(K) + \\ 5\text{tempo}(P) &= 0,6 + 1,44 + 5,5 = \mathbf{7,54s} \end{aligned}$$

$$\begin{aligned} \text{tempo}(T1.1) &= 5\text{tempo}(BB) + 12\text{tempo}(K) + \\ 5\text{tempo}(P) &= 0,6 + 1,44 + 5,5 = \mathbf{7,54s} \end{aligned}$$

#### 4.2.4 Configurar a posição de um objeto (T1.2)

Um objeto pode possuir mais de uma posição inicial, contudo, para o estudo de caso, está sendo levado em consideração que ele tenha apenas uma. A nova versão da ferramenta possui duas formas de posicionar o objeto, pode ser do mesmo jeito que é feito na ferramenta antiga ou pode ser utilizado a área de pré visualização para posicionar o objeto. Será contabilizado o segundo.

**Primeira versão:**

1. Posicionar e clicar no painel *Configuração Game Object* (P+BB);
2. Posicionar e clicar no campo de texto da posição do objeto (P+BB);
3. Digitar valor da posição (K);
4. Ir para o próximo campo de texto, pressionar TAB (K);
5. Digitar valor da posição (K);

**Nova versão:**

1. Posicionar e clicar no painel *Configuração gerais do objeto* (P+BB);
2. Posicionar e clicar (segurando) no objeto contido na área de pré visualização, em seguida liberar o objeto na posição que se deseja (P+BB);

$$\begin{aligned} \text{tempo}(T1.2) &= 2\text{tempo}(P) + 2\text{tempo}(BB) + 3\text{tempo}(K) = 2,2 + 0,2 + 0,36 = \mathbf{2,76s} \\ \text{tempo}(T1.2) &= 2\text{tempo}(P) + 2\text{tempo}(BB) = 2,2 + 0,2 = \mathbf{2,4s} \end{aligned}$$

#### 4.2.5 Configurar uma animação de um objeto (T1.3)

Configurar uma animação de um objeto é uma das ações mais importantes e demoradas da ferramenta, essa tarefa é responsável por definir o comportamento de um objeto quando um determinado acontece, além de ser responsável por estabelecer uma relação entre o objeto e o seu respectivo *spritesheet*.

**Primeira versão:**

1. Posicionar e clicar em *Configurações Game Object* (P+BB);
2. Posicionar e clicar no botão (+) ao lado de *Animações* (P+BB);
3. Posicionar e clicar em *Estado do objeto* (P+BB);
4. Posicionar e clicar em uma opção (P+BB);
5. Posicionar e clicar no botão (+) ao lado de *Estado do objeto* (P+BB);
6. Posicionar e clicar de *Sprite sheet* (P+BB);
7. Posicionar e clicar em uma opção de *Sprite sheet* (P+BB);
8. Identificar quadro inicial da animação no *spritesheet* (M);
9. Posicionar e clicar no campo da linha do quadro inicial (P+BB);
10. Digitar valor (M+K);
11. Posicionar no campo da coluna do quadro inicial através da tecla TAB (K);
12. Digitar valor (M+K);
13. Repetir os cinco últimos passos para o quadro final (3M+P+BB+3K);
14. Posicionar e clicar no campo de velocidade da animação (P+BB);
15. Digitar valor da velocidade da animação (M+3K);
16. Posicionar e clicar na opção *Flip vertical* (**Opcional**) (P+BB);
17. Posicionar e clicar na opção *Flip horizontal* (**Opcional**) (P+BB);
18. Posicionar e clicar em *Animation Policy* (P+BB);
19. Posicionar e clicar numa opção de *Animation Policy* (P+BB);
20. Posicionar e clicar no botão cadastrar (P+BB);

**Nova versão:**

1. Posicionar e clicar em *Animações do objeto* (P+BB);
2. Posicionar e clicar na lista com os *spritesheets* (P+BB);
3. Posicionar e clicar no botão *Frame inicial* (P+BB);
4. Posicionar e clicar no botão que representa o quadro inicial da animação (M+P+BB);
5. Posicionar e clicar no botão *Frame final* (P+BB);
6. Posicionar e clicar no botão que representa o quadro final da animação (M+P+BB);
7. Posicionar e clicar na aba *Propriedades* (P+BB);
8. Posicionar e clicar no campo *Velocidade* (P+BB);
9. Digitar o valor da velocidade (3K);
10. Posicionar e clicar no campo *Política* (P+BB);
11. Escolher uma opção (P+BB);
12. Posicionar e clicar na opção *Flip vertical* (**Opcional**) (P+BB);
13. Posicionar e clicar na opção *Flip horizontal* (**Opcional**) (P+BB);
14. Posicionar e clicar em uma opção do campo *Estados de animação* (P+BB);
15. Posicionar e clicar no botão *Salvar* (P+BB);

$$\text{tempo}(T1.3) = 15\text{tempo}(P) + 9\text{tempo}(K) + 15\text{tempo}(BB) + 7\text{tempo}(M) = 16,5 + 1,08 + 3 + 8,4 = \mathbf{28,98s}$$

$$\text{tempo}(T1.3) = 14\text{tempo}(P) + 3\text{tempo}(K) + 14\text{tempo}(BB) + 2\text{tempo}(M) = 15,4 + 0,36 + 2,8 + 2,4 = \mathbf{20,96s}$$

### 4.2.6 Configurar uma tela (T2)

A tarefa de configurar uma tela em ambas as versões da ferramenta é simples e tem seu tempo estimado facilmente através das seguintes sub tarefas:

1. Selecionar o plano de fundo (T2.1);
2. Criar um ou mais botões (T2.2)

Ou seja, o tempo que é levado para configurar uma tela é calculado através da seguinte equação  $tempo(T2) = tempo(T2.1) + N \times tempo(T2.2)$  seja  $N$  a quantidade de botões na tela.

De acordo com as secções 4.2.7 e 4.2.8, pode-se determinar que uma tela com dois botões é criada no seguinte tempo  $tempo(T2_{antiga}) = 5,3 + 2 \times 20,5 = \mathbf{46,3s}$  e  $tempo(T2_{nova}) = 5,3 + 2 \times 17,3 = \mathbf{39,9s}$

### 4.2.7 Configurar a textura da tela (T2.1)

O tempo que se leva para configurar a textura de uma tela do jogo, ou seja atribuir um plano de fundo a ela é dividido nas seguintes operações:

#### Primeira versão:

1. Posicionar e clicar em *Configuração Screen* (P+BB);
2. Posicionar e clicar em *Pesquisar* (P+BB);
3. Posicionar e clicar no arquivo de imagem que será usado como textura da tela (M+P+BB);
4. Posicionar e clicar em abrir (P+BB);

#### Nova versão:

1. Posicionar e clicar em *Configurações da tela* (P+BB);
2. Posicionar e clicar na lupa (P+BB);
3. Posicionar e clicar no arquivo de imagem que será usado como textura da tela (M+P+BB);
4. Posicionar e clicar em abrir (P+BB);

$$tempo(T2) = 3tempo(P) + 4tempo(BB) + tempo(M) = 3,3 + 0,8 + 1,2 = \mathbf{5,3s}$$

$$tempo(T2) = 3tempo(P) + 4tempo(BB) + tempo(M) = 3,3 + 0,8 + 1,2 = \mathbf{5,3s}$$



### 4.2.8 Criar um botão (T2.2)

A tarefa de criar um botão é baseada em três sub tarefas que são mostradas nas secções 4.2.9, 4.2.10 e 4.2.11. Logo, o tempo de criar um botão, é igual a soma dos tempos dessas sub tarefas  $tempo(T2.2) = tempo(T2.2.1) + tempo(T2.2.2) + tempo(T2.2.3)$ .

Substituindo pelos valores obtidos na respectivas secções, é possível verificar que para criar um botão leva-se o seguinte tempo (ver tabela 9):  $tempo(T2.2_{antiga}) = 20,5s$  e  $tempo(T2.2_{nova}) = 17,3s$ .

Tarefa	Tempo(s)	Tempo(s)
T2.2.1	11,5	11,5
T2.2.2	4,5	1,3
T2.2.3	4,5	4,5

Tabela 9 – Tempo que leva para executar as tarefas relacionadas a criação de um botão nas duas versões da ferramenta.

### 4.2.9 Configurar textura de um botão (T2.2.1)

Significa definir qual a imagem que deve ser exibida quando o botão da tela estiver sendo clicado ou não, é constituído das seguintes operações:

**Primeira versão:**

1. Posicionar e clicar em *Configurações Button Screen* (P+BB);
2. Posicionar e clicar em *Pesquisar* ao lado do campo referente a textura (P+BB);
3. Posicionar e clicar no arquivo de imagem que será usado como textura (M+P+BB);
4. Posicionar e clicar em abrir (P+BB);
5. Repetir os três últimos para a textura do botão quando for clicado (3P+M+3BB);

**Nova versão:**

1. Posicionar e clicar em *Configurações do botão* (P+BB);
2. Posicionar e clicar em *Pesquisar* ao lado do campo referente a textura (P+BB);
3. Posicionar e clicar no arquivo de imagem que será usado como textura (M+P+BB);
4. Posicionar e clicar em abrir (P+BB);
5. Repetir os três últimos para a textura do botão quando for clicado (3P+M+3BB);

$$\begin{aligned} tempo(T2.2.1) &= 7tempo(P) + 7tempo(BB) + \\ 2tempo(M) &= 7,7 + 1,4 + 2,4 = \mathbf{11,5s} \end{aligned}$$

$$\begin{aligned} tempo(T2.2.1) &= 7tempo(P) + 7tempo(BB) + \\ 2tempo(M) &= 7,7 + 1,4 + 2,4 = \mathbf{11,5s} \end{aligned}$$

**4.2.10 Posicionar um botão (T2.2.2)**

é a tarefa que consiste em posicionar um botão na tela que ele pertence, vale salientar que a segunda versão da ferramenta utiliza a área de pré visualização para fazer isso e, portanto, consome menos tempo do usuário, contudo o tempo é determinado pelas seguintes operações:

**Primeira versão:**

1. Posicionar e clicar em *Configurações Button Screen* (P+BB);
2. Posicionar e clicar em *Relativo X* (P+BB);
3. Digitar valor (3K);
4. Repetir os dois últimos para *Relativo Y* (P+BB+3K)

**Nova versão:**

1. Posicionar e clicar (segurando) no botão na área de pré visualização, em seguida soltar o botão onde se deseja que ele fique (P+BB);

$$\begin{aligned} \text{tempo}(T2.2.2) &= 3\text{tempo}(BB) + 3\text{tempo}(P) + 6\text{tempo}(K) = 0,6 + 3,3 + 0,72 = \mathbf{4,5s} \end{aligned}$$

$$\begin{aligned} \text{tempo}(T2.2.2) &= \text{tempo}(P) + \text{tempo}(BB) = 1,1 + 0,2 = \mathbf{1,3s} \end{aligned}$$

#### 4.2.11 Configurar o tamanho de um botão (T2.3)

É a tarefa que visa determinar o tamanho relativo que um botão ocupa na tela, consiste nas seguintes operações:

**Primeira versão:**

1. Posicionar e clicar em *Configurações Button Screen* (P+BB);
2. Posicionar e clicar em *Largura relativa* (P+BB);
3. Digitar valor (3K);
4. Repetir os dois últimos para *Altura relativa* (P+BB+3K)

**Nova versão:**

1. Posicionar e clicar em *Configurações do botão* (P+BB);
2. Posicionar e clicar em *Largura relativa* (P+BB);
3. Digitar valor (3K);
4. Repetir os dois últimos para *Altura relativa* (P+BB+3K)

$$\begin{aligned} \text{tempo}(T2.2.3) &= 3\text{tempo}(BB) + 3\text{tempo}(P) + 6\text{tempo}(K) = 0,6 + 3,3 + 0,72 = \mathbf{4,5s} \end{aligned}$$

$$\begin{aligned} \text{tempo}(T2.2.3) &= 3\text{tempo}(BB) + 3\text{tempo}(P) + 6\text{tempo}(K) = 0,6 + 3,3 + 0,72 = \mathbf{4,5s} \end{aligned}$$

Operação	Descrição	Tempo (s)
K	Digitar uma tecla, é usado para quando o usuário digita qualquer tecla seja a letra a ou a tecla SHIFT.	0,12
P	Posicionar o ponteiro do mouse em algum local da tela.	1,1
B	Pressionar ou liberar os botões do mouse	0,1
BB	Clique do mouse	0,2
H	Mover a mão do teclado para o mouse.	0,4
M	Pensar ou perceber algo relacionado ao sistema.	1,2

Tabela 10 – Operações e seus respectivos tempos para o sistema KLM.

### 4.3 Análise dos resultados

Após estimar o tempo que leva as principais tarefas envolvidas na construção de um jogo em ambas as versões da ferramenta do projeto CGT, é possível, com esses números, provar que o módulo de pré visualização tornou mais rápida e fácil a criação de jogos.

Para tanto, considera-se um jogo exemplo que tenha 20 objetos em mundos diferentes e 10 telas, apenas com os resultados obtidos nas secções 4.2.1 e 4.2.6 podemos calcular que esse jogo na primeira versão da ferramenta terá seus objetos e telas configurados em aproximadamente **1h 17min**, enquanto que a nova versão teria o mesmo jogo construído em aproximadamente **57min**. Vale lembrar que esse tempo da ferramenta antiga pode aumentar se incluirmos o tempo das execuções do jogo que o usuário faz a medida que vai configurando esses objetos. Ver tabela 11.

Pode ser considerado então que o aproveitamento obtido através da implementação da nova ferramenta, foi de pelo menos **25,97%**, ou seja, pode-se assumir que os usuários criarão jogos 25,97% mais rápidos. Como forma de ilustrar esse resultado, pode-se pensar que com quatro jogos criados na antiga versão da ferramenta, seria criado um quinto, caso fosse utilizado a ferramenta nova.

<b>Ferramenta 1.0</b>	1h 17min
<b>Ferramenta 2.0</b>	57min

Tabela 11 – Resultado de análise KLM para ambas as versões da ferramenta.

## 5 Conclusão e trabalhos futuros

Neste trabalho, apresentou-se melhorias significativas para a ferramenta de criação de jogos do projeto CGT que tem como objetivo tornar a criação de jogos ainda mais fácil, lógica e produtiva. De tal forma que, os usuários não tenham necessidade de possuir conhecimentos técnicos na criação de jogos, aumentando ainda mais o alcance da ferramenta ao público alvo. Além disso, a ferramenta torna-se ideal quando existe uma necessidade de rapidez na demanda ou se, essa demanda, é um jogo simples que na mão das grandes produtoras de jogo custaria bem mais que um jogo criado pela ferramenta. Ou seja, quando a necessidade é provar um conceito ou demonstrar uma ideia a ferramenta é uma excelente escolha.

Com o avanço e crescimento da indústria de jogos, ferramentas acessíveis e de confiança, farão a diferença nesse mercado onde o diferencial consiste, principalmente, na criatividade do autor do jogo. Então, para que o projeto CGT continue competitivo e interessante é importante investir na experiência do usuário e, dessa forma, ganhar espaço nessa área e, como grande pilar posto desde o surgimento do projeto, aumentar o alcance dos usuários menos especializados contanto apenas com as ideias deles para produzir jogos cada vez melhores.

Posto isso, ainda existem melhorias quem podem ser feitas em prol de uma ferramenta ainda mais robusta e eficiente. Assim, para trabalhos futuros seria interessante considerar os seguintes pontos:

- a) Disponibilizar mais ações na área de pré visualização que, atualmente, conta apenas com a função de mover os objetos para novas posições na tela, seria interessante incluir outras propriedades e ações (modificar o tamanho do objeto, por exemplo). Ou seja, transformar a área de pré visualização no verdadeiro protagonista da ferramenta.
- b) Possibilitar a pré visualização de mensagens exibidas no jogo, tais como as mensagens de vitória e derrota no mundo, assim como o contador que é inserido quando o jogo é do tipo “sobreviver durante um período de tempo”.
- c) Prover de meios para a inserção rápida de telas e botões, a ferramenta

carrega imagens externa para esses dois componentes, entretanto, muitas vezes, necessita-se incluir algo bem simples como uma mensagem de texto. Seria interessante a ferramenta ter configurações expressas para mensagens do jogo.

- d) Possibilitar a criação de outros tipos de jogos, por exemplo, jogos de corrida e jogos em três dimensões que contem com outra perspectiva de câmera.
- e) Outro ponto que a ferramenta poderia disponibilizar aos produtores de jogos, seria meios para inserir anúncios nos jogos construídos. Sabe-se que a publicidade em jogos destinados a dispositivos móveis, por exemplo, representa boa parte dos ganhos com o jogo. Então se a ferramenta permitir isso de forma natural, der opções ao produtor do jogo, atrairia ainda mais usuários.

O desenvolvimento da ferramenta deve continuar evoluir prezando sempre pelos princípios estabelecidos pelo projeto CGT, de tal forma que, o fato de o projeto ser código aberto, pode auxiliar bastante com o crescimento dele.

# Referências

AQUINO, V. G. de. *Ceará Game Tools: Uma ferramenta de software livre para geração automática de games*. 79 p. Monografia (Bacharel em Engenharia de Computação) — Instituto Federal de Educação, Ciência e Tecnologia do Ceará, Fortaleza, 2015. Citado 4 vezes nas páginas 8, 9, 32 e 49.

CARD, S. K.; MORAN, T. P. The keystroke-level model for user performance time with interactive systems. *Xerox Palo Alto Research Center*, v. 23, n. 7, p. 396–410, 1980. Citado na página 32.

CGT. *Ceará Game Tools: Sobre o Projeto*. 2015. Disponível em: <<http://www.cgt.ifce.edu.br/sobre.php>>. Acesso em: 15.11.2015. Citado na página 8.

CGT. *Repositório do código fonte do projeto Ceará Game Tools hospedado na plataforma Github*. 2015. Disponível em: <<https://github.com/hexat/projetocgt>>. Acesso em: 9.12.2015. Citado na página 68.

KIERAS, D. Using the keystroke-level model to estimate execution times. *University of Michigan*, v. 1, 2001. Citado na página 33.



# Anexos

# ANEXO A – Manual da ferramenta

Após as melhorias implementadas na nova versão da ferramenta, foi preciso atualizar o manual existente em trabalhos anteriores (AQUINO, 2015). Usou-se a mesma estrutura do manual anterior, atualizando-o para adequar-se às novas telas implementadas. Então, com este complemento será possível encontrar uma descrição sucinta da ferramenta CGT(Ceará Game Tools), mostrando todas as suas funcionalidades e imagens que explicam as suas configurações. Além disso, com o intuito de mostrar as funcionalidades que a ferramenta possui, será demonstrada a criação de um jogo com a ferramenta CGT.

## A.1 Criar tela

Uma tela representa uma tela do jogo inicial, podendo ser encontrada como uma tela de configurações, uma tela de créditos, entre outros exemplos. Para criar uma tela basta ir no menu: Editar→Adicionar→Tela da ferramenta, ou o atalho: ALT+T (ver figura 18). No painel de configuração de uma tela, o usuário pode configurar o plano de fundo e o áudio. No exemplo a seguir foi selecionada uma imagem para o plano de fundo e adicionado botões a ela, figura 19.

Agora serão configurados os botões que foram adicionados a tela. 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 tela 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 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 tela e os valores devem variar de 0 a 1, representando a porcentagem da tela. A área de pré-visualização permite arrastar o botão para a posição desejada, atualizando o painel de configuração em seguida (figura 20). Com isso, a ferramenta permite ao usuário criar telas de acordo com a necessidade do jogo e com os botões que lhe forem preciso.

## A.2 Criar mundo

O botão de uma tela pode levar 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 ir no menu: Editar→Adicionar→Mundo da ferramenta, ou o atalho: ALT+M (ver figura 21). Como mostrado na imagem 22, o mundo possui algumas características gerais, que são: nome, plano de fundo e música. Além disso, configurações de vitória, derrota e pausa do jogo.

### A.2.1 Critérios de vitória

Na figura 23, temos as configurações do critério de vitória para um mundo que podem ser cinco (eliminar inimigos, sobreviver, pegar bônus, completar percurso e atingir pontuação) e da mensagem que será exibida quando o jogador vencer. Para selecionar um dos critérios basta marcá-lo, notar que podem haver mais de um critério para o mundo.

- a) **Eliminar inimigos**: Será atingido quando todos os inimigos que forem destruíveis forem derrotados.
- b) **Sobreviver**: Configurar qual o valor(inteiro) de score deve ser atingido para que o critério seja atingido, ver imagem 27.
- c) **Pegar bônus**: Será atingido quando todos os bônus existentes tiverem sido consumidos pelo ator.
- d) **Completar percurso**: 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, ver figura 28
- e) **Atingir pontuação**: Informar o tempo (valor inteiro em segundos) que o personagem deve sobreviver para que o critério seja atingido, ver figura 29.

### A.2.2 Critérios de derrota

Na figura 24, temos as configurações de derrota desse mundo, os critérios podem ser: ator morrer ou contagem regressiva, uma mensagem de derrota também

pode ser configurada.

- a) **Ator morrer:** Será atingido quando o personagem possuir vida igual ou menor que zero.
- b) **Contagem regressiva:** Deve-se 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, ver imagem 30.

Além dessas mensagens de derreta e vitória, na figura 25, é possível configurar a mensagem que será exibida quando o mundo estiver em pausa. Por fim, na figura 26, tem-se o painel de configuração da câmera do mundo.

### A.2.3 Configurar mensagens

A configuração das mensagens são todas iguais, então será demonstrada apenas a configuração um deles. Para configurar será necessário ajustar posicionamento, dimensão, imagens e botões. Uma mensagem 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, as imagens de borda horizontal e canto inferior direito são imagens usadas para se definir a borda de toda a janela da mensagem. Ver figura 31.

Para adicionar botões na mensagem basta clicar no botão destacado na figura 32. 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 tela 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 tela.

## A.3 Configurar *spritesheet* dos objetos

Para adicionar um *spritesheet* de um objeto basta ir no menu Editar→Spritesheet→Adicionar da ferramenta. A janela da figura 33 será aberta, então o usuário deve escolher uma imagem PNG e informar o número de linhas e colunas.

## A.4 Ator principal

Para cada mundo existirá um ator principal que corresponde ao personagem que será controlado por quem estiver jogando. Ele possui algumas características que são comuns a outros objetos, tais como o inimigo, opositor e bônus.

A animação é uma propriedade que todos esses objetos tem em comum e significa o que é mostrado na tela quando algo acontece com o objeto (andar, ficar parado, morrer) é feita através de um *spritesheet* que é uma imagem que possui várias outras imagens dentro que possibilitam que uma animação seja configurada. Então, deve ser escolhido um *spritesheet* (figura 33), isso pode ser feito através do menu Editar→Spritesheet→Adicionar ou pelo botão que existe no painel de configuração, depois disso, deve ser feita as demais configurações da animação.

Ao escolher na aba de animações do objeto o *spritesheet* correspondente a esse objeto, todos os *sprites* que compõem a folha dele são exibidos abaixo e, com eles, é possível definir o *frame* inicial e final da animação. Para isso, basta clicar no botão correspondente do *frame* em que se deseja configurar e então clicar no *sprite* correspondente, a ferramenta informa ao lado do botão através de um texto qual o *sprite* selecionado, ver figura 34.

O próximo passo é configurar as propriedades dessa animação que, até agora, possui apenas um *frame* inicial e final. Das propriedades que devem ser configuradas temos (figura 35):

- a) **Velocidade**: representa de quanto em quanto tempo o *frame* atual é trocado pelo próximo.
- b) **Política**: indica como se deve alternar entre os *frames* nessa animação, os valores podem ser:
  - **Normal**: as imagens são mostradas na sequência, do *frame* inicial ao final uma única vez;
  - **Reversivo**: as imagens são mostradas na ordem inversa uma única vez;
  - **Loop**: as imagens são mostradas na ordem normal repetidas vezes;
  - **Loop Reversivo**: as imagens são mostradas na ordem inversa repetidas vezes;

- **Loop PingPong**: as imagens são mostradas na ordem normal depois na ordem inversa repetidas vezes;
  - **Loop Random**: as imagens são mostradas em qualquer ordem.
- c) **flip horizontal ou vertical**: indica se deve ser modificado a imagem aplicando nela uma rotação.
- d) **Estado**: representa em que momento deve ser exibido a animação, por exemplo: parado, andando para esquerda, dano, morto, olhando para cima.

Após escolhido os quadros inicial e final e as propriedades, a animação pode ser salva com o botão de salvar no painel, com isso, a animação deverá constar na lista de animações do ator que fica na terceira aba do painel (ver figura 36). Nessa lista é possível adicionar ou editar uma animação.

No que foi configurado as animações do ator é preciso configurar as propriedades gerais do objeto, que consistem na informação da vida desse objeto, além de suas posições iniciais e velocidade de movimento, ver figura 37. Vale salientar que, assim que configurado as posições iniciais, o objeto é posicionado na área de pré visualização, já que é nesse momento que a ferramenta sabe onde deve colocá-lo. Em seguida, deve ser configurado as dimensões do objeto, ou seja, tamanho dele e o tamanho do espaço de colisão (38). Ainda nas configurações do objeto, é preciso definir os sons que o objeto faz em duas ocasiões: colisão e eliminação (ver figura 39).

Até então, as propriedades que foram configuradas são comuns aos objetos: ator, inimigos, opositores, bônus e projétil. Portanto, por fim, devem ser configuradas as configurações exclusivas do ator que são mostradas na figura 40. Deve ser selecionado o projétil do ator, caso ele possua, o tempo de recuperação e as ações dele, que é um mapeamento da entrada que é dada pelo jogador com a ação que o personagem deve executar.

## A.5 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 41).

## A.6 Configurar inimigos

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. 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 42).

## A.7 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*, ver figura 43.

## A.8 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 44).

## A.9 Configurar HUD

HUD são elementos de informação que irão ser exibidos na tela, por exemplo vida, munição, tempo, pontuação, e outros. No jogo criado foi configurado apenas uma barra de vida 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 45).

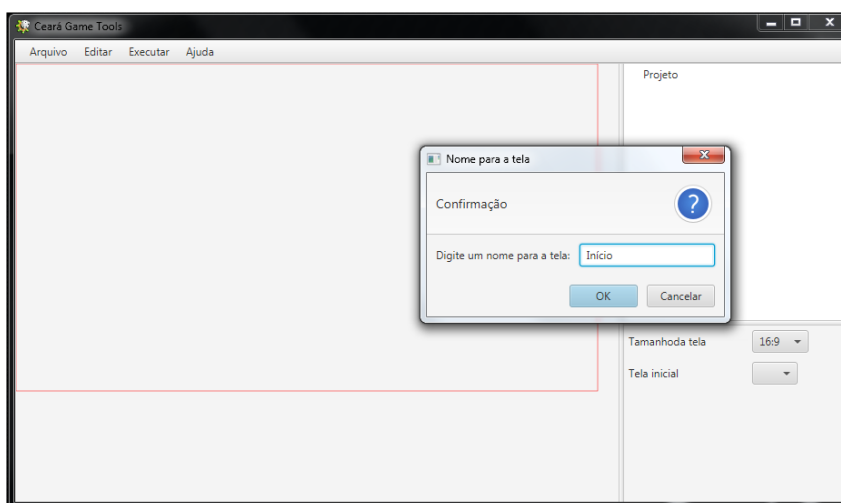


Figura 18 – Inserindo uma nova tela na ferramenta.

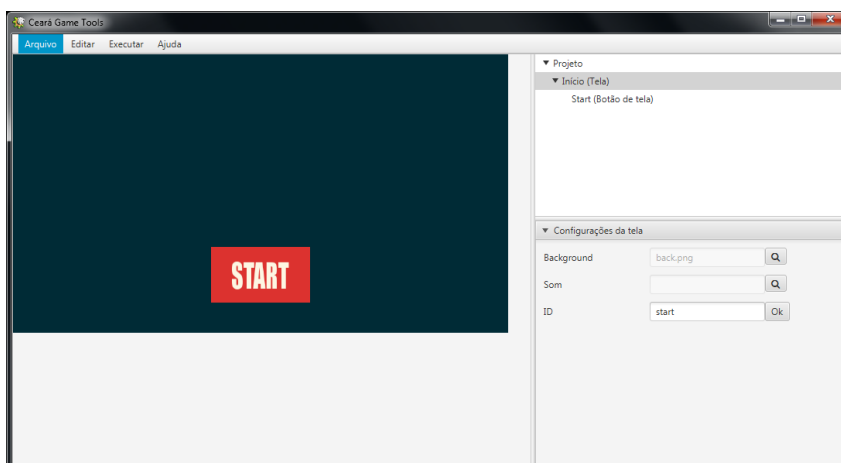


Figura 19 – Tela de um jogo exemplo sendo configurada.



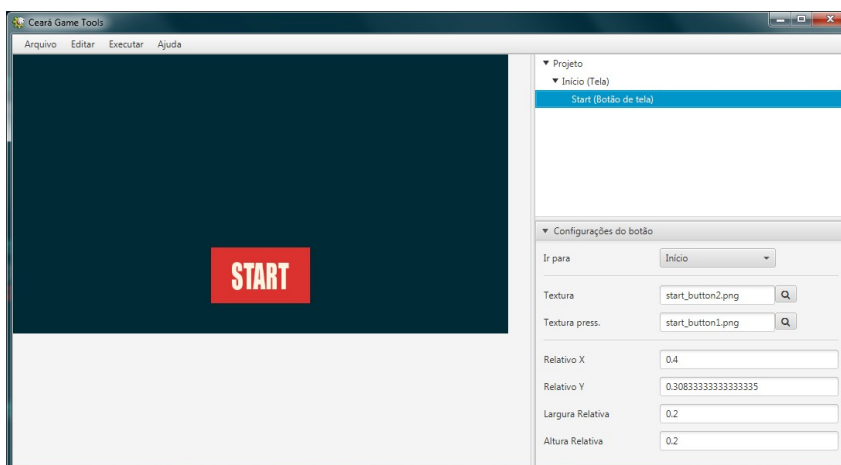


Figura 20 – Configurando um botão em tela de exemplo do jogo.

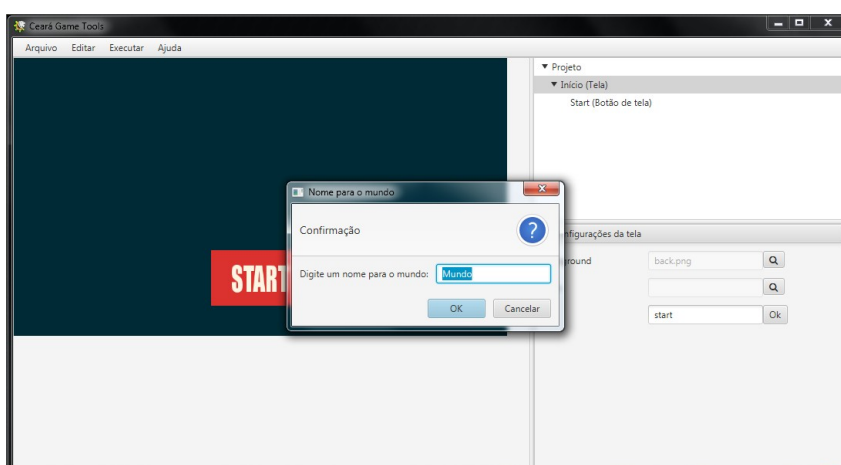


Figura 21 – Adicionando um mundo na nova versão da ferramenta

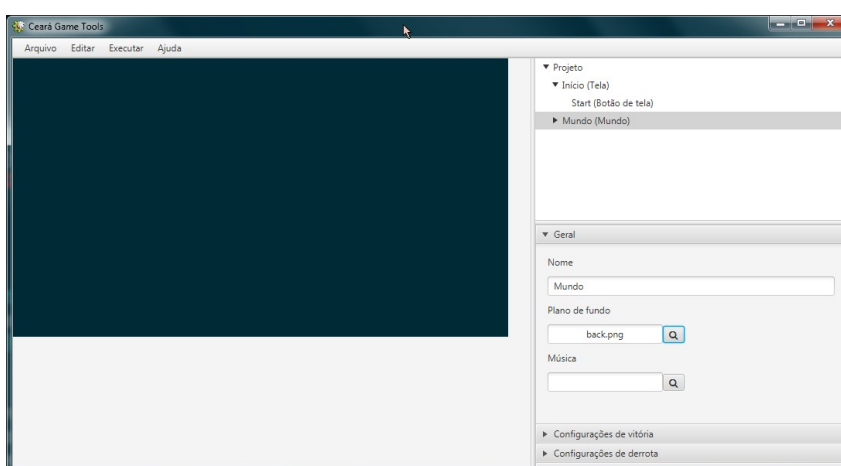


Figura 22 – Propriedades gerais de um mundo.

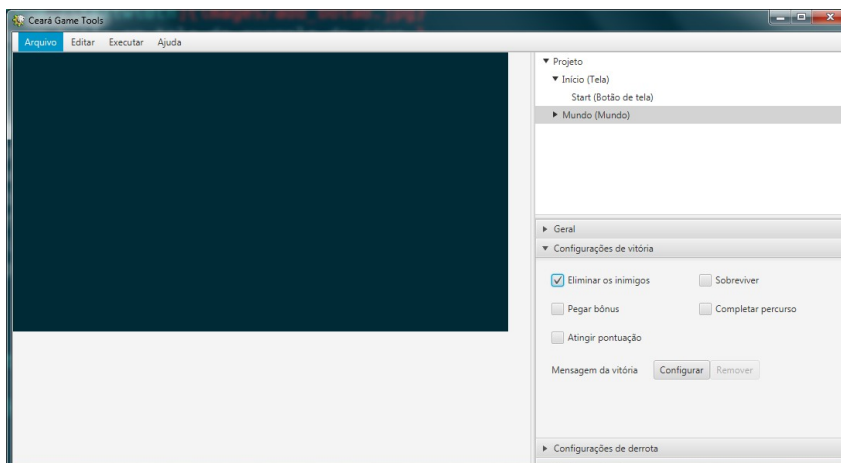


Figura 23 – Propriedades de vitória do mundo.

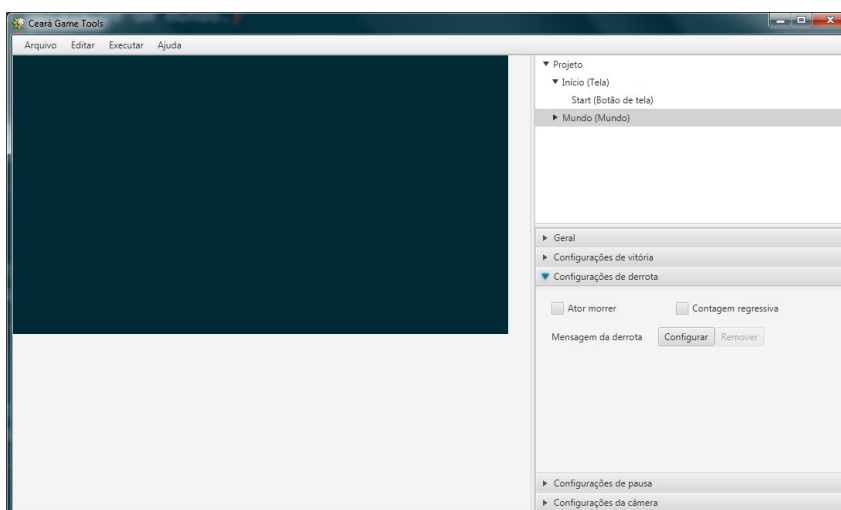


Figura 24 – Propriedades de derrota do mundo.

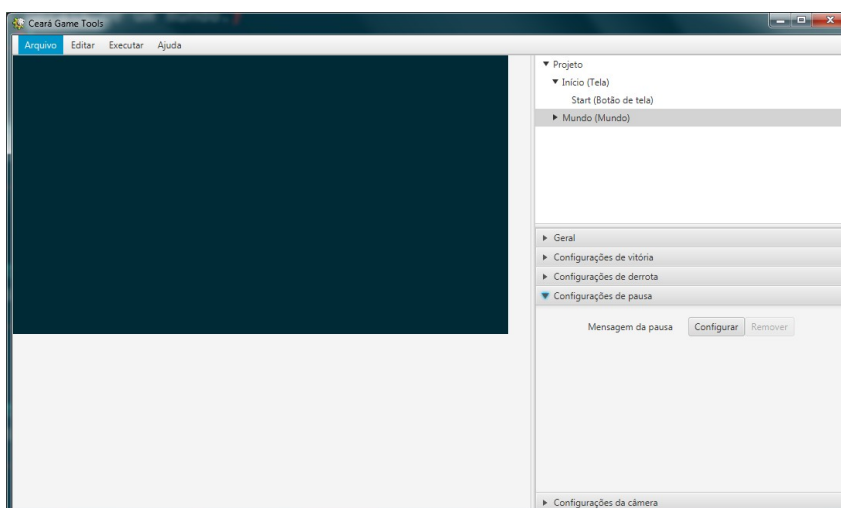


Figura 25 – Propriedades da pausa do mundo.

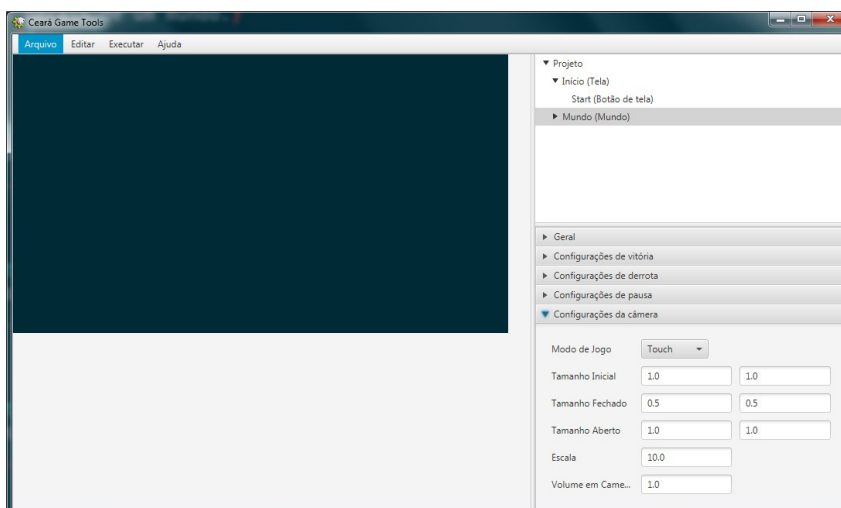


Figura 26 – Propriedades da câmera do mundo.

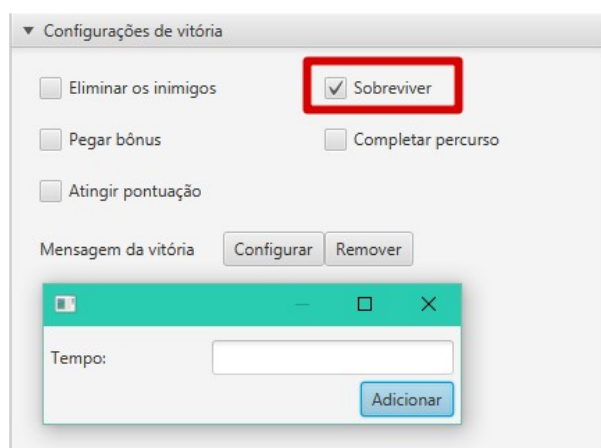


Figura 27 – Configurações do critério de vitória sobreviver no jogo.

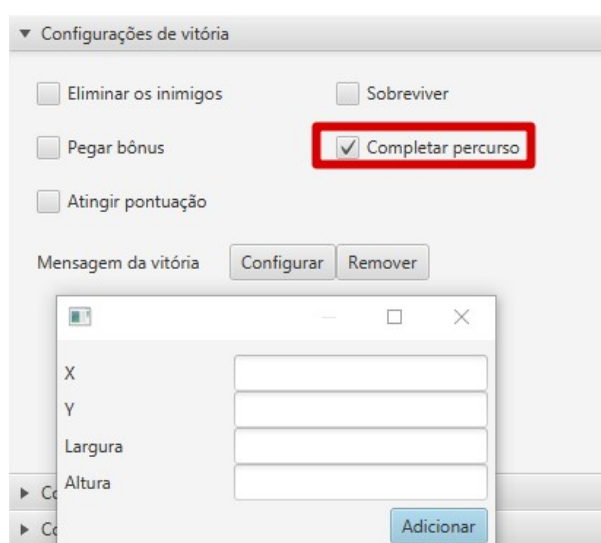


Figura 28 – Configurações do critério de vitória completar percurso no jogo.

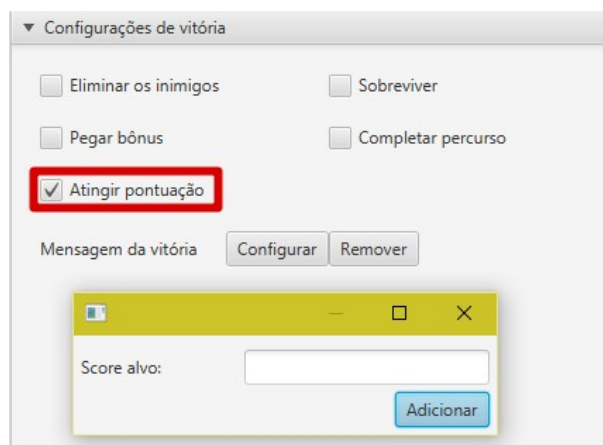


Figura 29 – Configurações do critério de vitória de atingir uma pontuação no jogo.

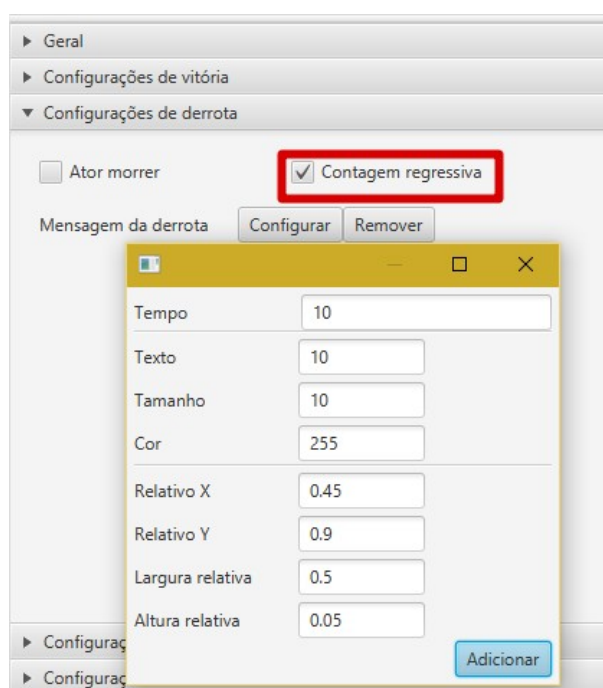


Figura 30 – Configurações do critério de derrota de contagem regressiva no jogo.

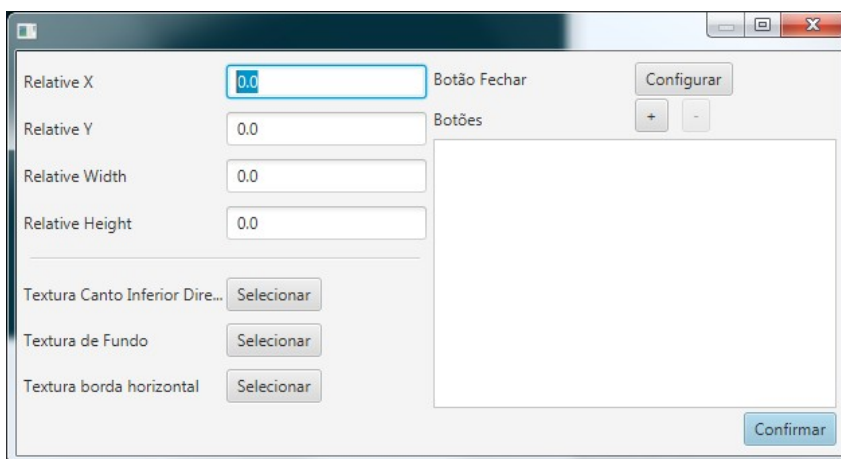


Figura 31 – Janela de configuração de uma mensagem no mundo, seja ela para vitória, derrota ou pausa.

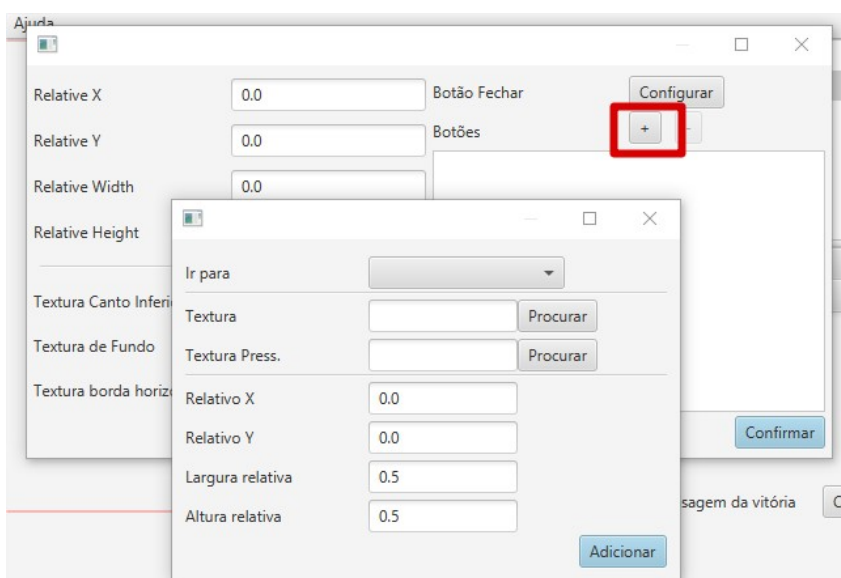


Figura 32 – Configuração dos botões de uma mensagem do mundo.

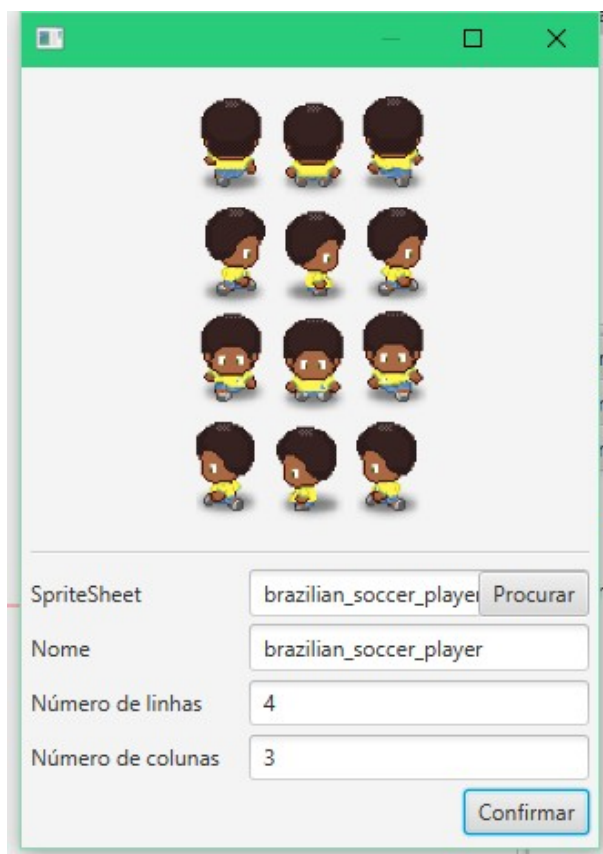


Figura 33 – Janela para adicionar *spritesheets* na ferramenta.

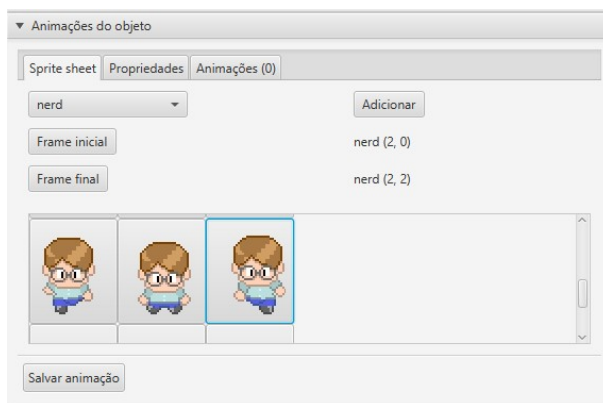


Figura 34 – Painel de configuração da animação de um objeto

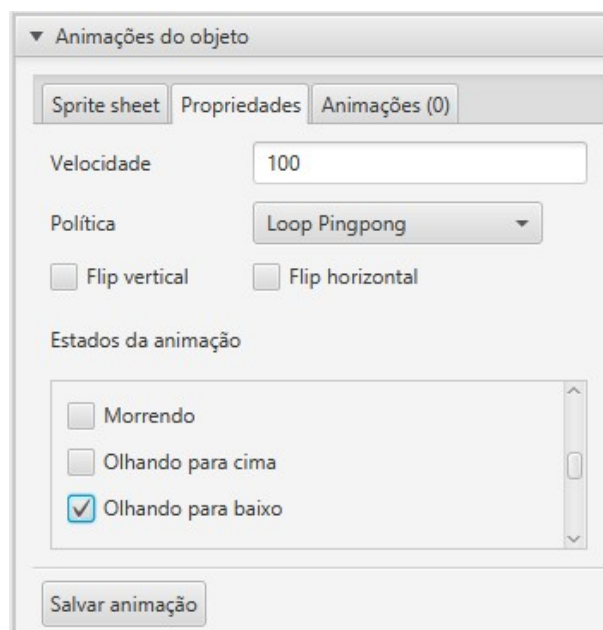


Figura 35 – Propriedades de uma animação nas configurações de um objeto.

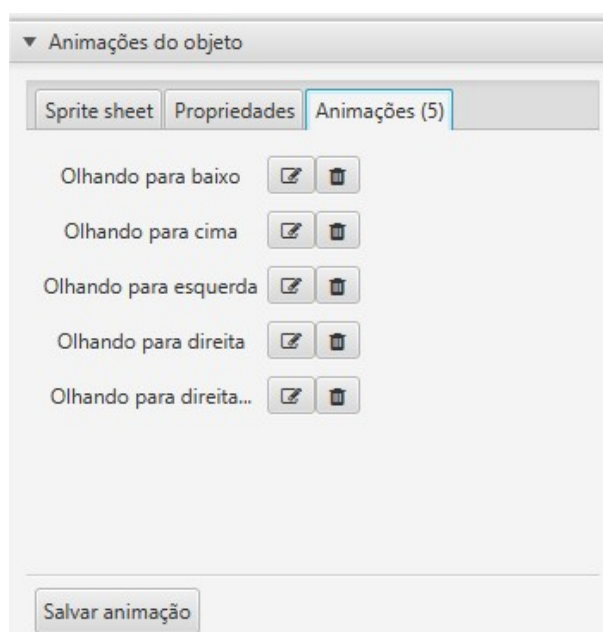


Figura 36 – Lista de animações de um objeto do jogo

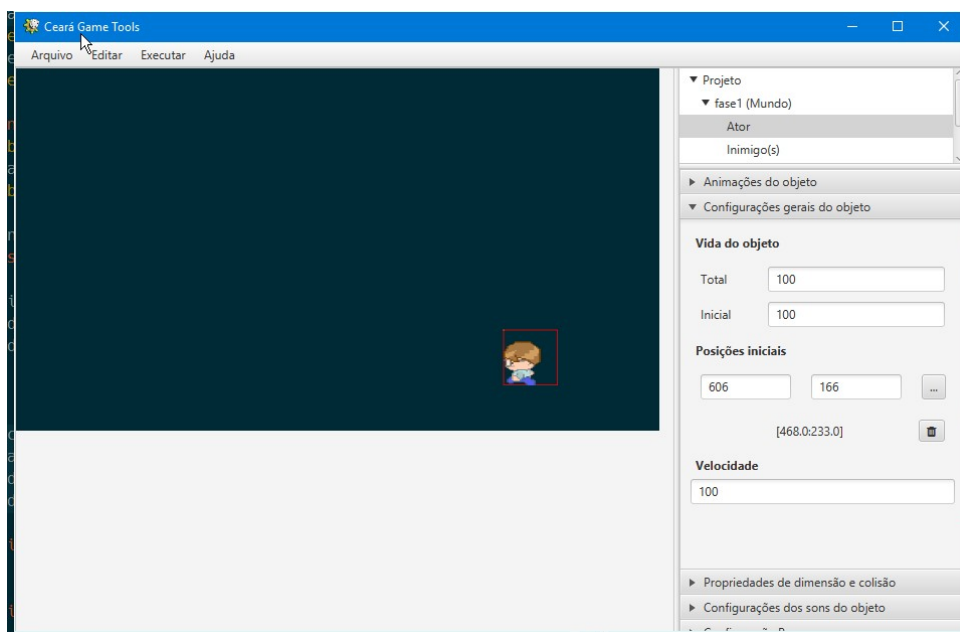


Figura 37 – Propriedades gerais de um objeto

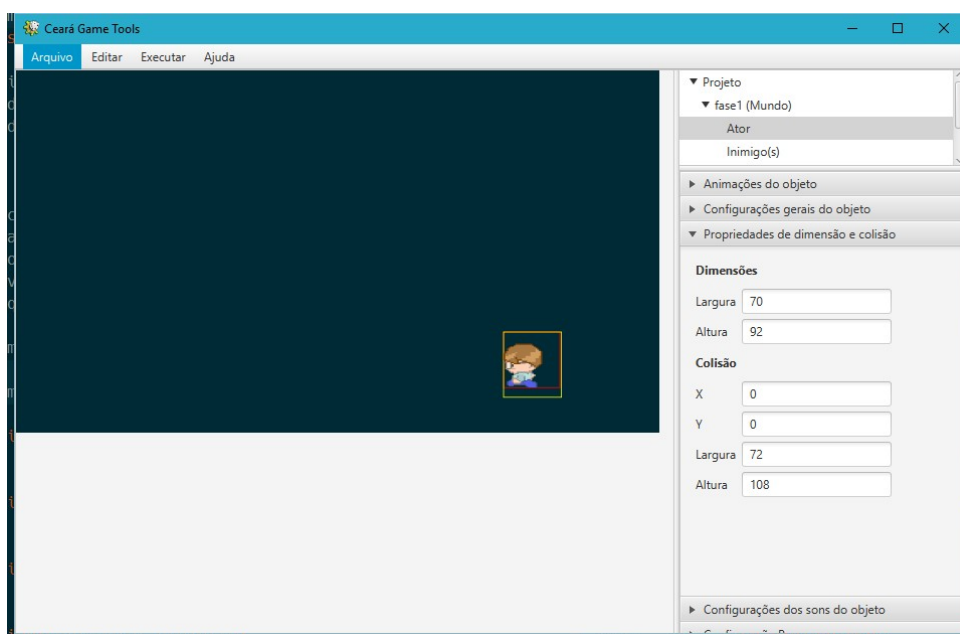


Figura 38 – Propriedades de dimensões e colisão de um objeto no jogo.



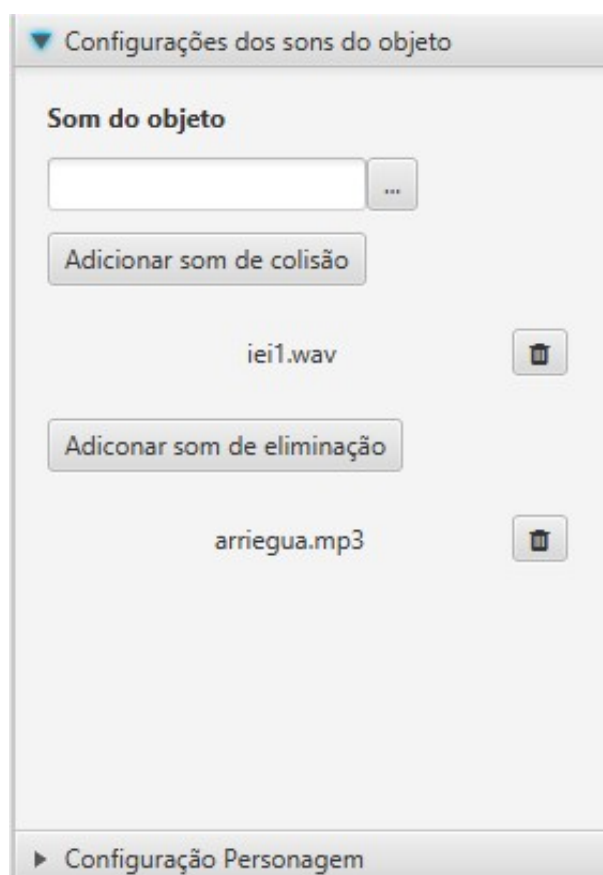
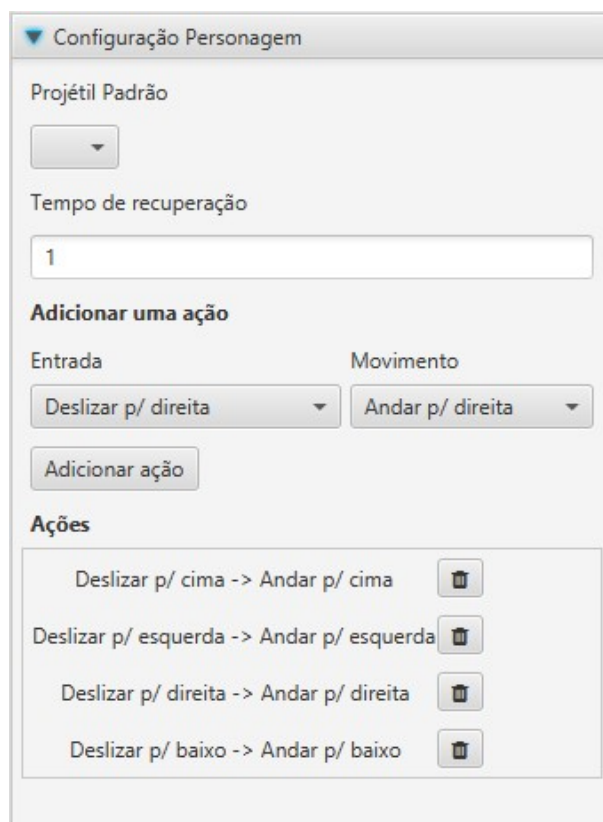


Figura 39 – Configuração dos sons que um objeto faz.



Configuração Personagem

Projétil Padrão

Tempo de recuperação

1

**Adicionar uma ação**

Entrada Movimento

Deslizar p/ direita Andar p/ direita

Adicionar ação

**Ações**





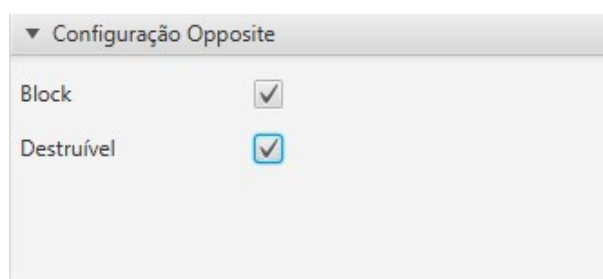
Deslizar p/ cima -> Andar p/ cima	
Deslizar p/ esquerda -> Andar p/ esquerda	
Deslizar p/ direita -> Andar p/ direita	
Deslizar p/ baixo -> Andar p/ baixo	

Figura 40 – Propriedades exclusivas do ator do jogo



Configuração Opposite

Block	<input checked="" type="checkbox"/>
Destruível	<input checked="" type="checkbox"/>

Figura 41 – Configurações de um opositor.

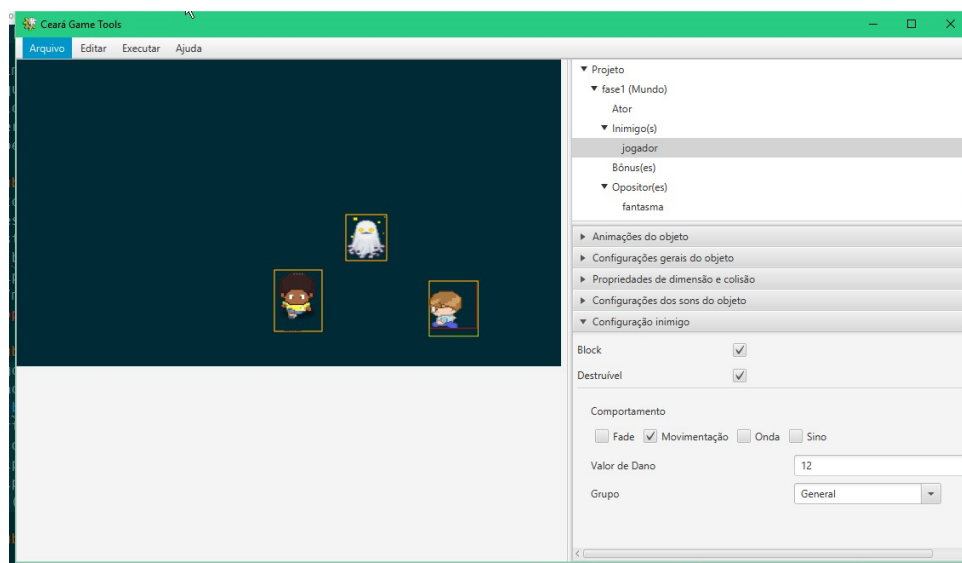


Figura 42 – Propriedades do inimigo

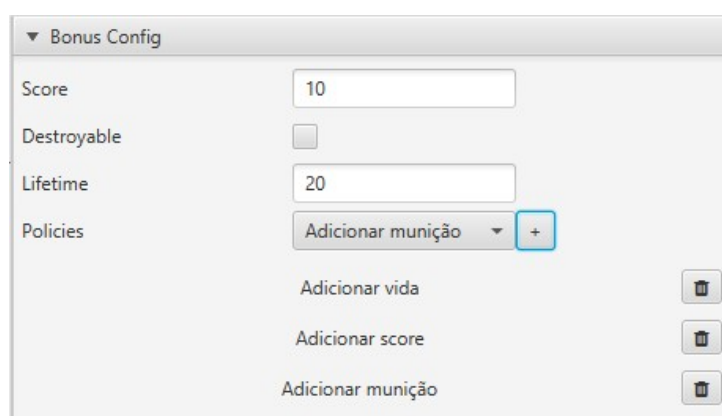


Figura 43 – Propriedades de um bônus do jogo

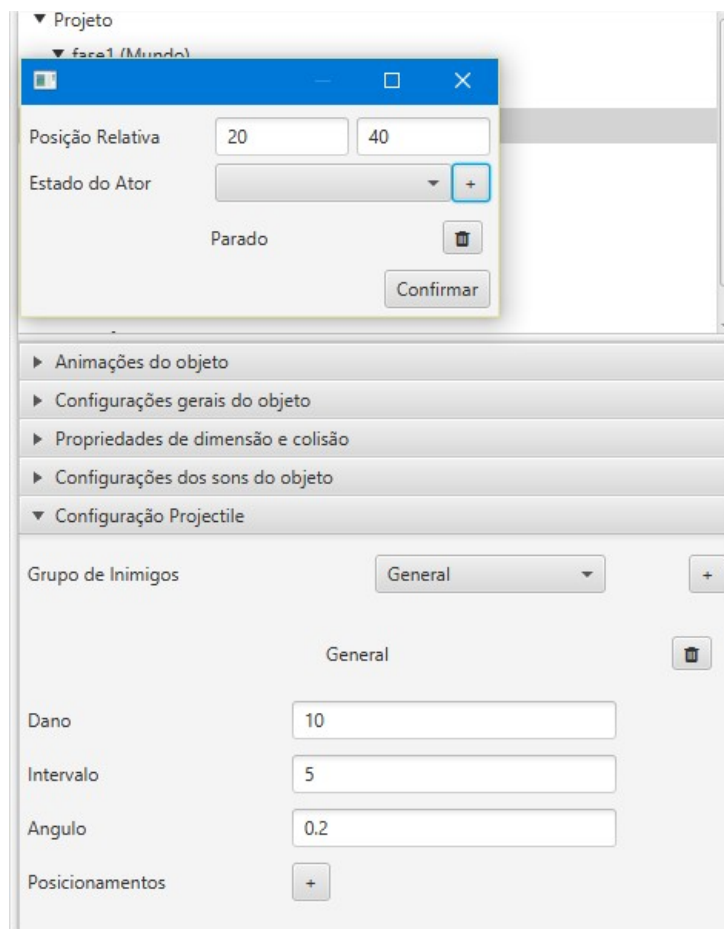


Figura 44 – Propriedades de um projétil no jogo.

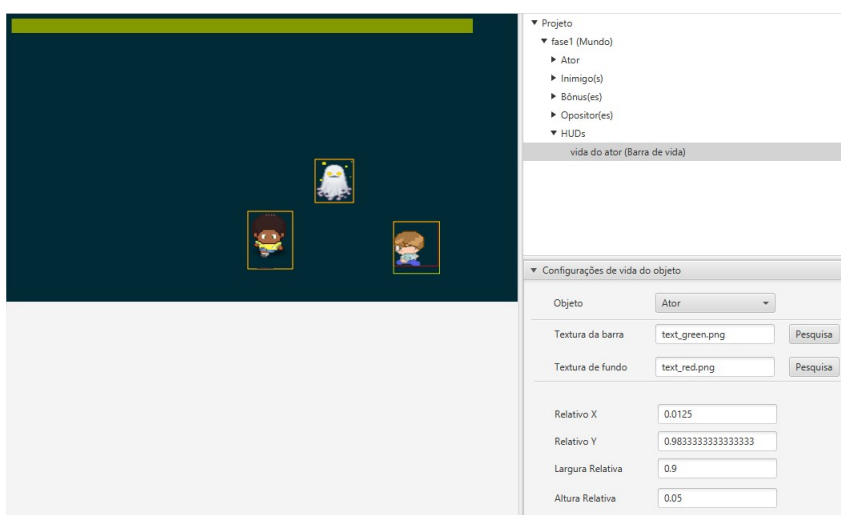


Figura 45 – Propriedades da barra de vida de um objeto.

## ANEXO B – Diagramas de classes

Dessa página em diante, pode-se ver o diagrama de classes do módulo que foi e implementado juntamente com esse trabalho, os diagramas são das classes presentes na ferramenta contidas no projeto CGT no pacote `br.edu.ifce.cgt.application.vo` e outras classes que precisaram ser adaptadas para a nova versão. Os diagramas foram gerados a partir da ferramenta *Enterprise Architect* que lê o código fonte e gera os diagramas UML para ele, obteve-se o código fonte a partir do repositório *Git* do projeto (CGT, 2015b). As demais classes do projeto, são irrelevantes para o que foi proposto aqui ou não foram alteradas para as implementações que foram feitas para a nova versão da ferramenta.

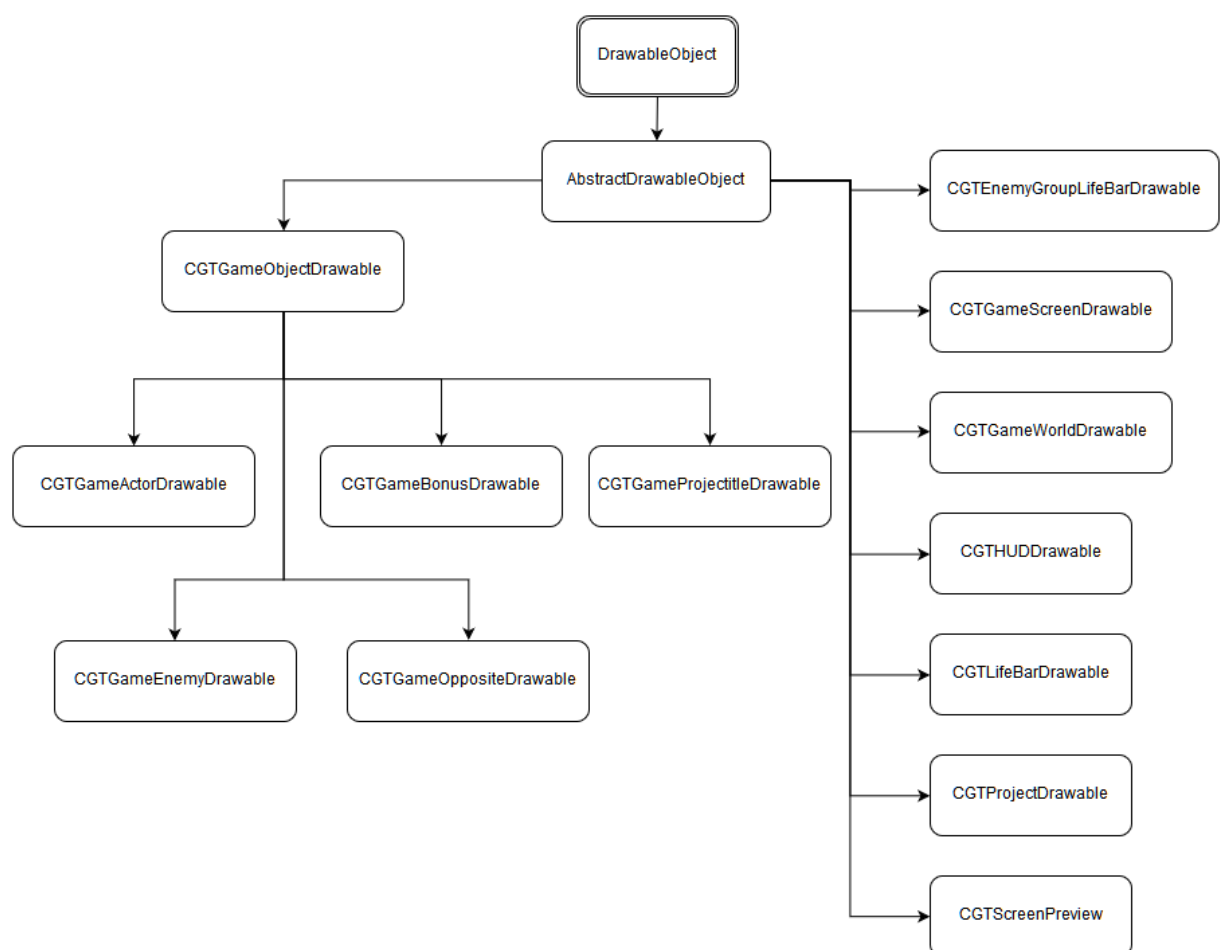
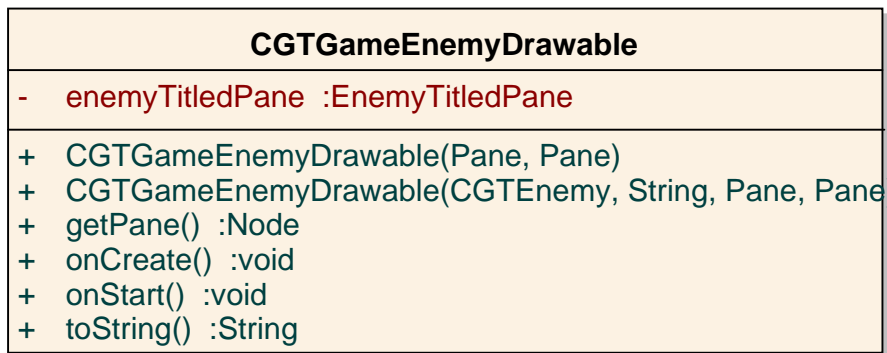
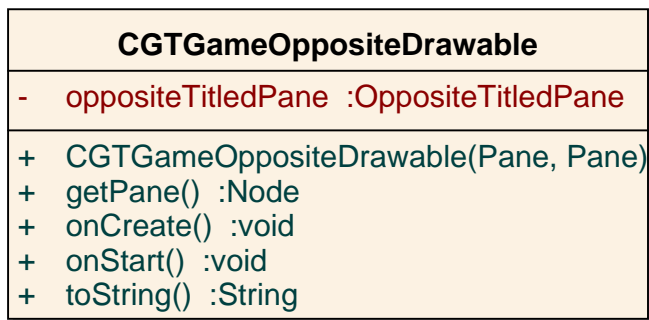
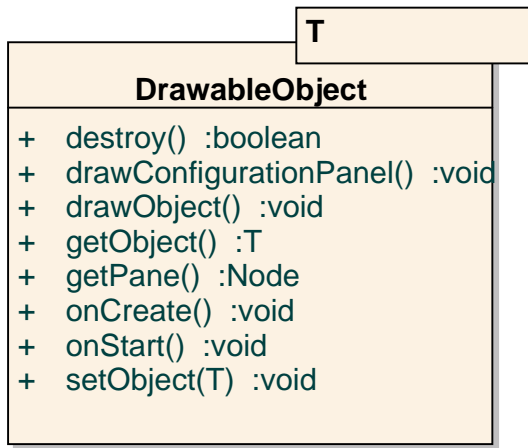


Figura 46 – Classes envolvidas no módulo de pré visualização.



CGTGameActorDrawable
- actorTitledPane :ActorTitledPane
+ CGTGameActorDrawable(Pane, Pane) + CGTGameActorDrawable(CGTActor, String, Pane, Pane) + getPane() :Node + onCreate() :void + onStart() :void + toString() :String

CGTGameProjectileDrawable
- pane :ProjectileTitledPane
+ CGTGameProjectileDrawable(Pane, Pane) + CGTGameProjectileDrawable(CGTProjectile, String, Pane, Pane) + getPane() :Node + onCreate() :void + onStart() :void + toString() :String

CGTHUDDrawable
- lifes :ArrayList<CGTLifeBarDrawable> - name :String
+ CGTHUDDrawable(Pane, Pane, String) + destroy() :boolean + drawConfigurationPanel() :void + drawObject() :void + getLifes() :ArrayList<CGTLifeBarDrawable> + getName() :String + getPane() :Node + onCreate() :void + onStart() :void + toString() :String

CGTLifeBarDrawable
<ul style="list-style-type: none"> <li>- life :IndividualLifeBar</li> <li>- lifePane :ConfigLifePane</li> <li>- preview :Draggable = new Draggable()</li> </ul>
<ul style="list-style-type: none"> <li>+ CGTLifeBarDrawable(Pane, Pane)</li> <li>+ destroy() :boolean</li> <li>+ drawConfigurationPanel() :void</li> <li>+ drawObject() :void</li> <li>+ getDraggable() :Draggable</li> <li>+ getLife() :IndividualLifeBar</li> <li>+ getPane() :Node</li> <li>+ onCreate() :void</li> <li>+ onStart() :void</li> <li>+ setSizeLife() :void</li> <li>+ toString() :String</li> </ul>

	T
AbstractDrawableObject	
<ul style="list-style-type: none"> <li>- drawableConfigurationsPane :Pane</li> <li>- drawableObjectPane :Pane</li> <li>- object :T</li> </ul>	
<ul style="list-style-type: none"> <li>+ AbstractDrawableObject(Pane, Pane)</li> <li>+ AbstractDrawableObject(T, Pane, Pane)</li> <li>+ getDrawableConfigurationsPane() :Pane</li> <li>+ getDrawableObjectPane() :Pane</li> <li>+ getObject() :T</li> <li>+ setObject(T) :void</li> <li>+ updateConfigPane(Pane) :void</li> <li>+ updateConfigPane(Node) :void</li> <li>+ updateDrawPane(Node) :void</li> <li>+ updateDrawPaneClear(Node) :void</li> </ul>	



CGTEnemyGroupLifeBarDrawable	
-	<b>life</b> :EnemyGroupLifeBar <b>lifePane</b> :ConfigGroupLifePane <b>preview</b> :Draggable = new Draggable()
+	CGTEnemyGroupLifeBarDrawable(Pane, Pane) destroy() :boolean drawConfigurationPanel() :void drawObject() :void getDraggable() :Draggable getLife() :EnemyGroupLifeBar getPane() :Node onCreate() :void onStart() :void setSizeLife() :void toString() :String

CGTGameScreenDrawable	
-	<b>screenPane</b> :ConfigScreenPreviewPane
+	CGTGameScreenDrawable(Pane, Pane) CGTGameScreenDrawable(Pane, Pane, int, int) CGTGameScreenDrawable(CGTScreen, Pane, Pane, int, int) CGTGameScreenDrawable(CGTScreen, Pane, Pane) destroy() :boolean drawConfigurationPanel() :void drawObject() :void getPane() :Node onCreate() :void onStart() :void toString() :String

CGTButtonScreenPreview	
-	buttonPane :ConfigButtonPreviewPane name :String preview :Draggable = new Draggable() screenName :String
+	CGTButtonScreenPreview(Pane, Pane) CGTButtonScreenPreview(CGTButtonScreen, String, Pane, Pane) destroy() :boolean drawConfigurationPanel() :void drawObject() :void getImage() :Draggable getPane() :Node getScreenName() :String onCreate() :void onStart() :void setSizeButton() :void toString() :String

		<i>T &gt; CGTGameObject</i>
CGTGameObjectDrawable		
-	bounds :Rectangle collision :Rectangle gameObjectTitledPane :GameObjectPane preview :Draggable = new Draggable() worldName :String	
+	CGTGameObjectDrawable(Pane, Pane) CGTGameObjectDrawable(T, String, Pane, Pane) destroy() :boolean drawConfigurationPanel() :void drawObject() :void getDraggable() :Draggable getObjectPane() :GameObjectPane getPane() :Node getWorldName() :String onStart() :void setSizeObject() :void setWorldName(String) :void showGameObjectDialog() :Optional<Pair<String, String>>	

	<i>BorderPane</i>
<b>PreviewPane</b>	
<ul style="list-style-type: none"> <li>+ <u>DESKTOP_JAR_PATH</u> :String = "desktop/deskto... {readOnly}</li> <li>+ <u>DESKTOP_ZIP_PATH</u> :String = "desktop/deskto... {readOnly}</li> <li>+ drawableConfigurationsPane :Pane</li> <li>+ drawableObjectPane :Pane</li> <li>- openRecentMenu :Menu</li> <li>- rootItem :CGTProjectDrawable</li> <li>- running :boolean</li> <li>- tree :TreeView&lt;DrawableObject&gt;</li> </ul>	
<ul style="list-style-type: none"> <li>+ about() :void</li> <li>+ addBonus() :void</li> <li>+ addButtonScreen() :void</li> <li>+ addEnemy() :void</li> <li>+ addEnemyLifeBar() :void</li> <li>+ addGearInformation() :void</li> <li>+ addObjectLifeBar() :void</li> <li>+ addOpponent() :void</li> <li>+ addProjectile() :void</li> <li>+ addScreen() :void</li> <li>+ addSpriteSheet() :void</li> <li>+ addWorld() :void</li> <li>+ beforeClosing() :void</li> <li>+ closeProject() :void</li> <li>- copyDesktopFiles() :void</li> <li>+ editSpriteSheet() :void</li> <li>+ exit() :void</li> <li>+ exportProject() :void</li> <li>- getActorWorldNode(String) :TreeItem&lt;DrawableObject&gt;</li> <li>- getHUDNode(String) :TreeItem&lt;DrawableObject&gt;</li> <li>- getScreenNode(String) :TreeItem&lt;DrawableObject&gt;</li> <li>- getWorldNode(String) :TreeItem&lt;DrawableObject&gt;</li> <li>- isWin() :boolean</li> <li>- localDefaultDirectory() :String</li> <li>+ newProject() :void</li> <li>- open(File) :void</li> <li>+ openProject() :void</li> <li>+ PreviewPane()</li> <li>- runDesktop() :void</li> <li>+ runProject() :void</li> <li>+ saveProject() :void</li> <li>+ saveProjectAs() :void</li> <li>- setupTree() :void</li> <li>- showValidateDialog(List&lt;CGTError&gt;) :void</li> <li>- updateRecent() :void</li> <li>- updateTree() :void</li> </ul>	