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

Agradecimentos

Resumo

O crescimento da industria 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

Lista de tabelas

Sumário

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 foma 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 inciativa do professor do IFCE Carlos Hairon Ribeiro Gonçalves apoiado pelo CNPQ[1] 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. 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*).

[1] Centro Nacional de desenvolvimento científico e tecnológico.

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 sí 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.

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

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 *Windows*, *Linux* e *MacOS*.
`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 *Android*.
`ios`	Permite exportar o jogo para execução nos dispositivos que rodam o sistema operacional *iOS*.

Tabela 1 – Módulos existentes no projeto CGT.

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.

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*[2] do objeto e preenchendo campos de texto com o valor da linha e coluna do *sprite* correspondente a animação configurada (ver imagem 2).

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

[2] Imagem que contém todas as faces de um objeto no jogo.

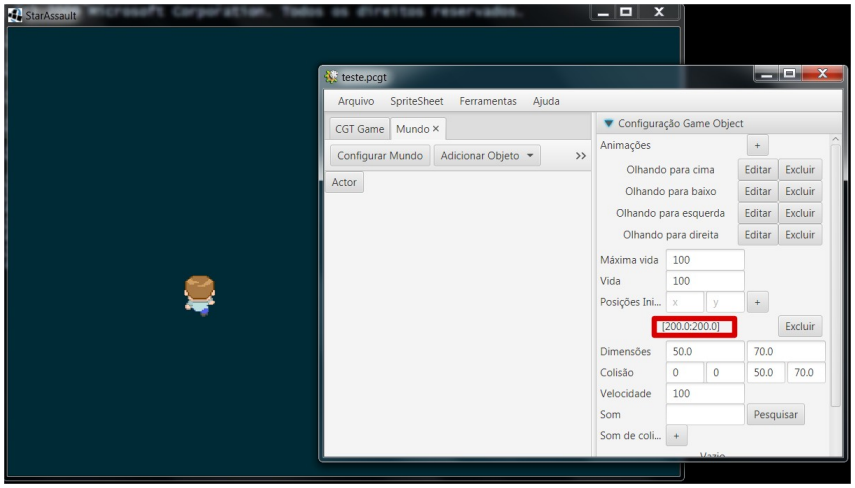


Figura 1 – Problema de pré-visualização na `ferramenta 1.0`.

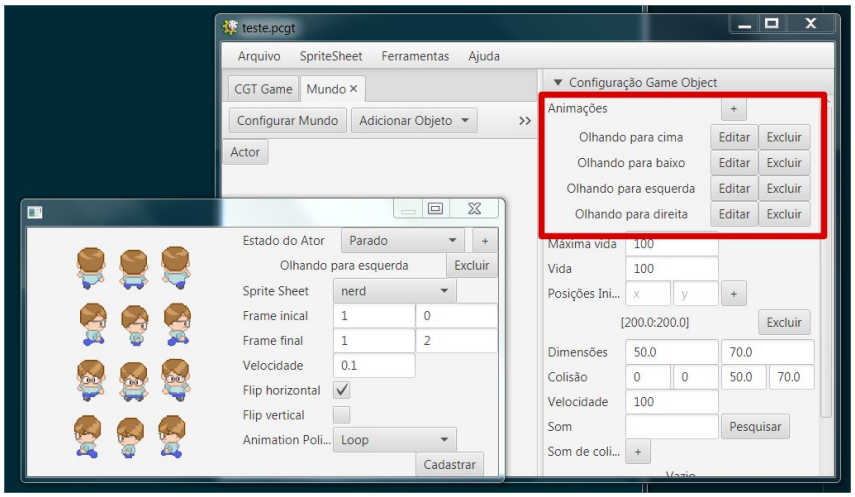


Figura 2 – Problema com os controles na configuração de uma animação na `ferramenta 1.0`.

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 4, 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.

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 na secção de *downloads*[1].

[1] http://www.cgt.ifce.edu.br/downloads.php

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.

2.2.1 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.

Menu	No menu é possível realizar ações que refletem no projeto todo, por exemplo: salvar, adicionar *spritesheet*, executar o jogo.
Abas	As abas representam os agregadores de objetos, podem conter a aba inicial do projeto, um *mundo* ou uma tela.
Toolbar	Permite realizar ações referentes ao agregador, se for de um mundo possibilitar adicionar um inimigo ao jogo, por exemplo.
Botões de objetos	Cada objeto é representado por um botão que pertencem a uma determinada aba.
Painel lateral	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.

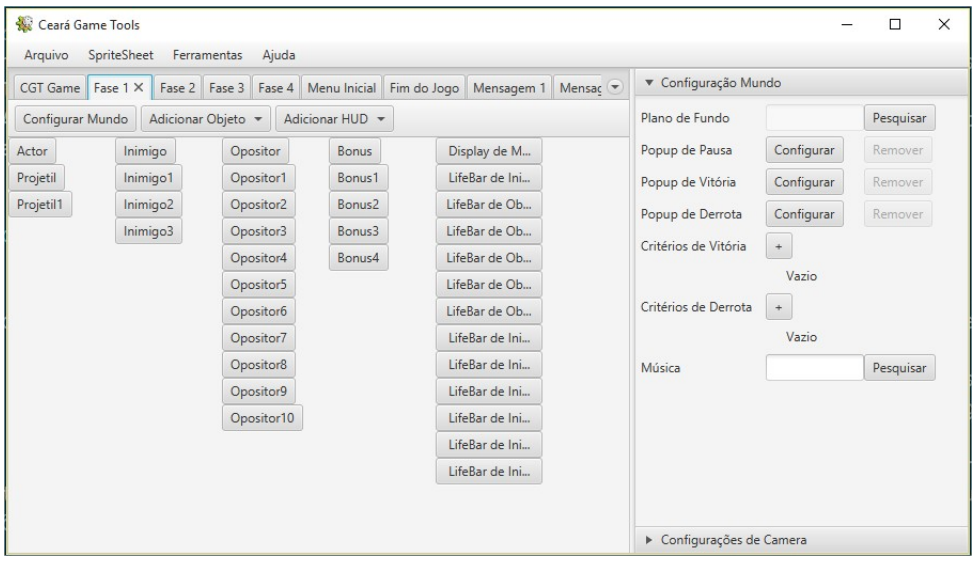


Figura 3 – Disposição dos objetos na versão anterior da ferramenta.

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 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.

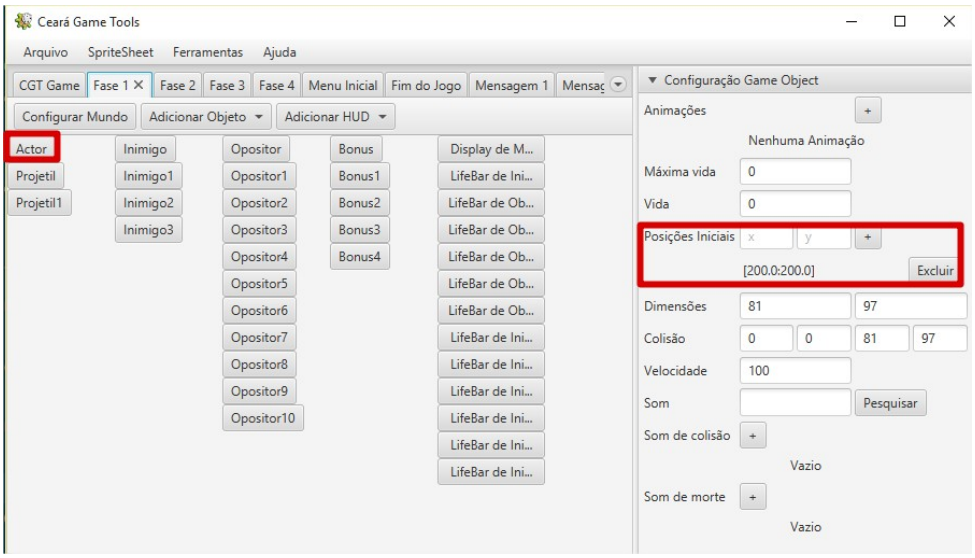


Figura 4 – Configuração da posição inicial de um objeto na primeira versão da ferramenta.

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

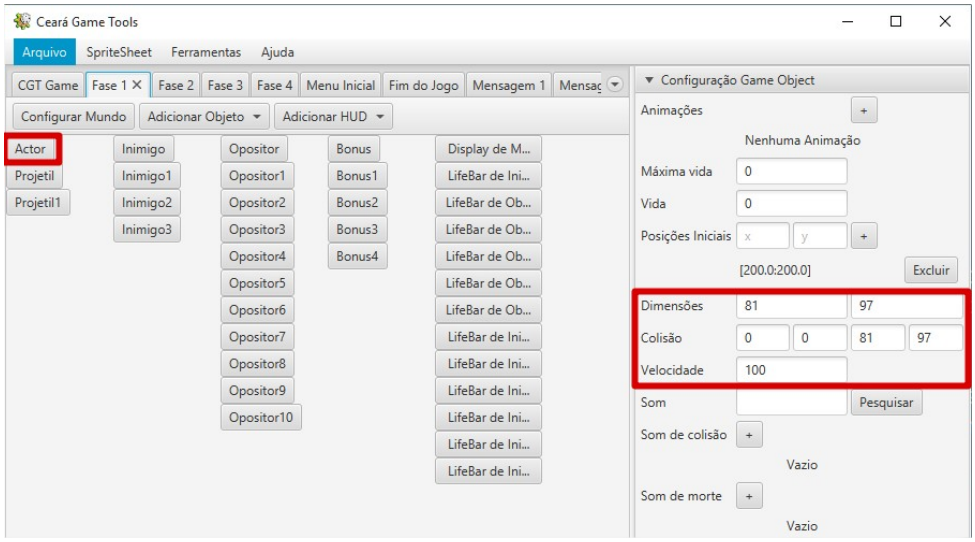


Figura 5 – Configuração das dimensões de um objeto na primeira versão da ferramenta.

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 (*spritesheet*), 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.

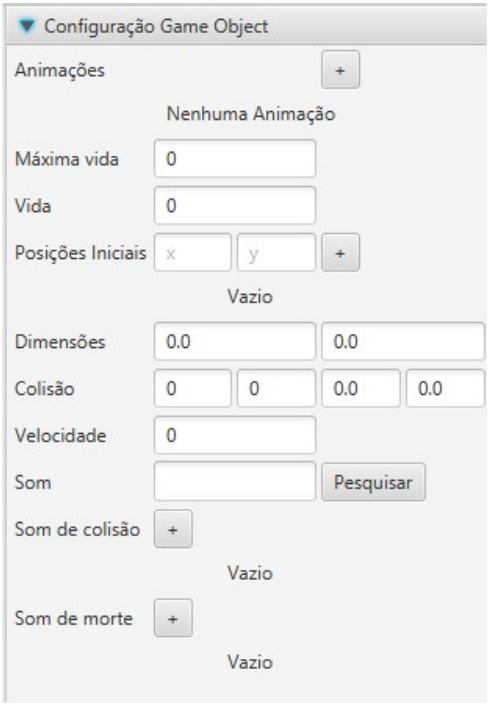


Figura 7 – Painel de configuração de um objeto do jogo, existe para os seguintes objetos: ator, inimigo, bônus, opositor e projétil.

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

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

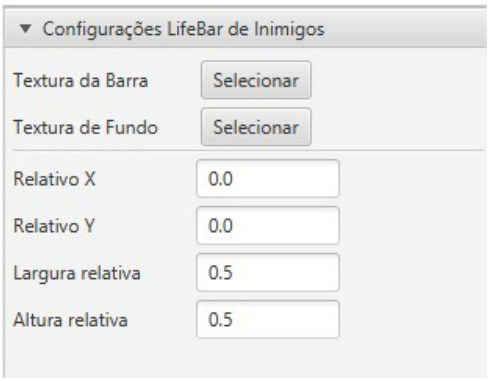


Figura 10 – Painel de configuração da barra de vida de um objeto do jogo.

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 4).

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, consequentemente, 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.

Menu	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 *toolbar* e possui atalhos para facilitar o uso. Por exemplo, `CTRL + I` para adicionar um inimigo ao jogo.
Área de pré visualização	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.
Árvore de objetos	Componente que sumariza todos os objetos existentes no jogo. A árvore é responsável por exibir os objetos em sua hierarquia.
Painel de configuração	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 disso.

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*.

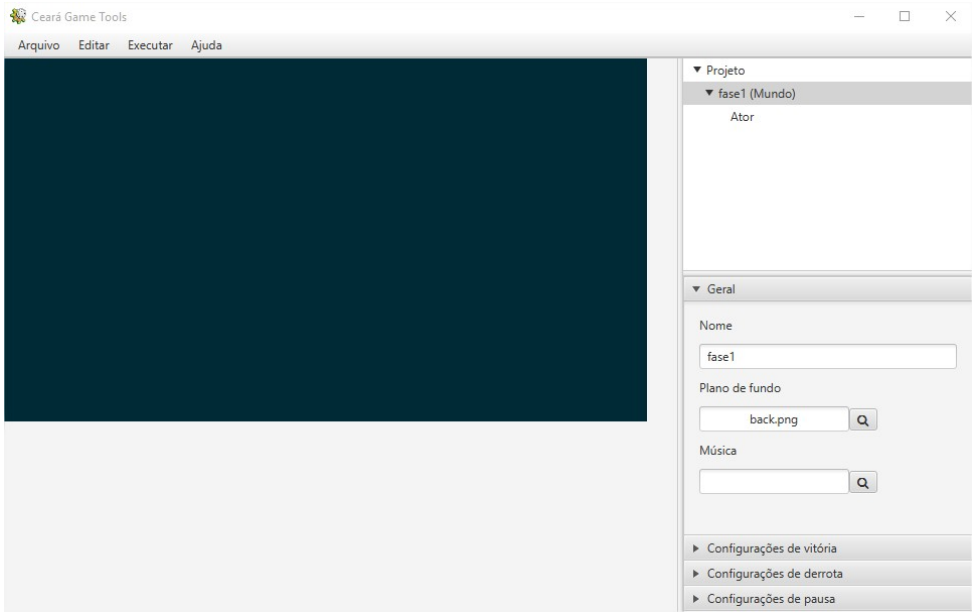


Figura 11 – Nova tela inicial da ferramenta com os novos controles que possibilitaram as melhorias.

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 `DrawbleObject`. A interface `DrawbleObject` 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 `DrawbleObject`. 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 `DrawbleObject` (`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
Projetil	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, consequentemente, 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.

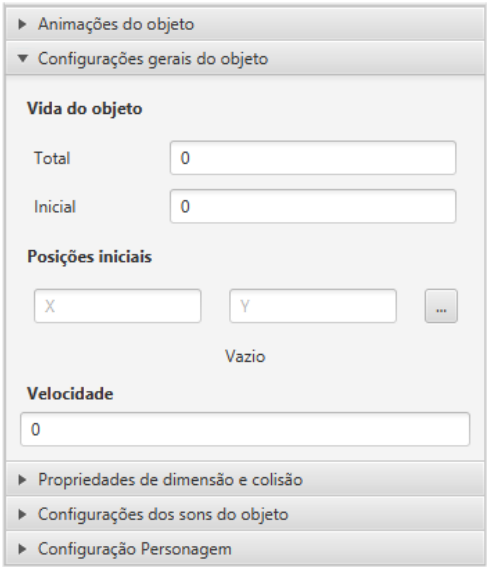


Figura 13 – Painel de configuração de um ator no jogo

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 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 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

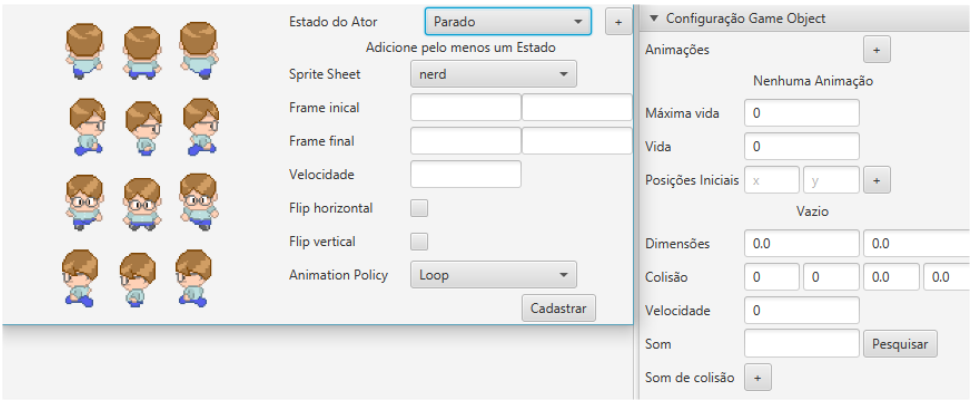


Figura 14 – Janela para adicionar animações para um objeto na primeira versão da ferramenta.

necessário apenas clicar no quandro 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.

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 capitulo, 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

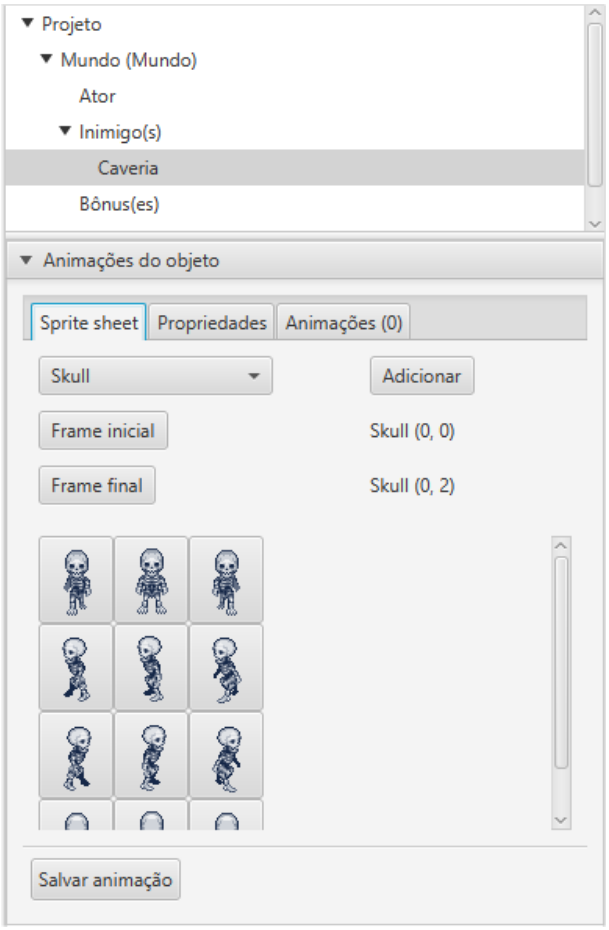


Figura 15 – Painel de configuração das animações do objeto na nova versão da ferramenta.

(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.

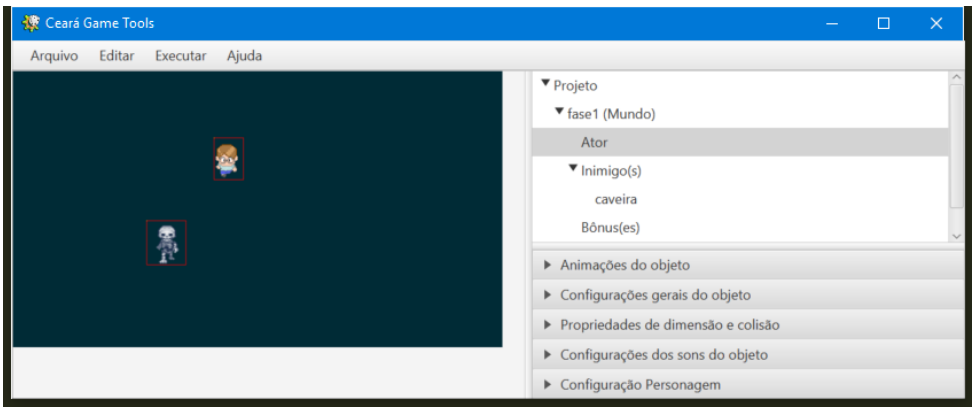


Figura 16 – Tela da ferramenta mostrando a área de pré visualização com dois objetos configurados.

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 especificas 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 secção 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.

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.

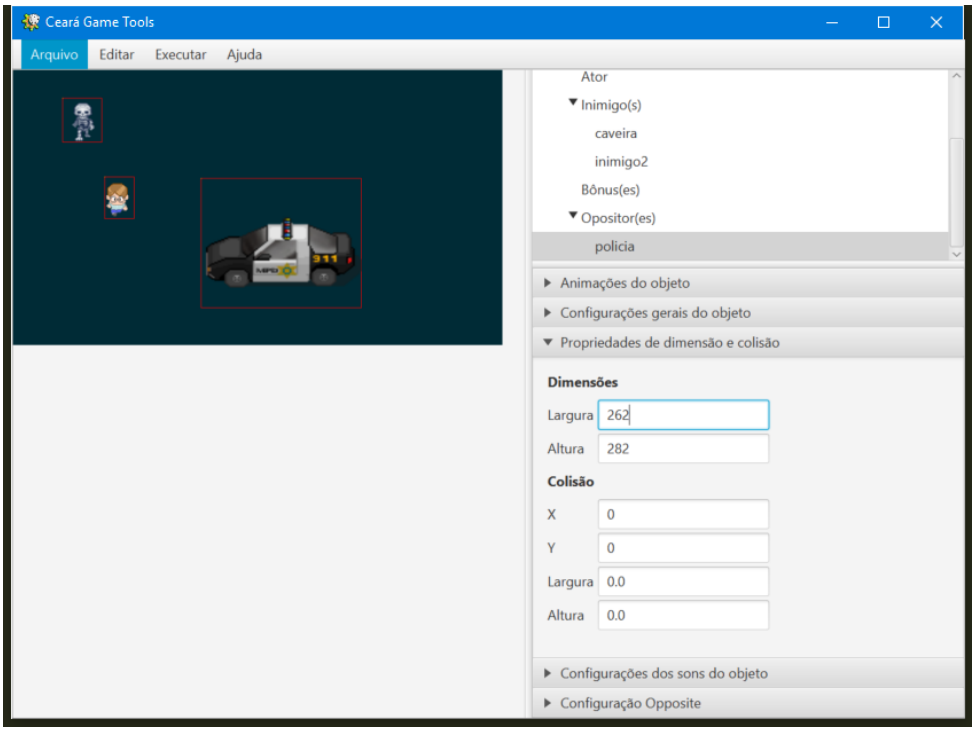


Figura 17 – Exemplo de dimensão do objeto sendo percebida no momento que é alterada na configuração

```
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();
      }
   }});
```

4 Estudo de caso

Neste capitulo 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.

4.1 Análise dos resultados

Ao colocar o módulo de pré visualização na ferramenta, exclui-se a necessidade de frequentemente executar o jogo para verificar qualquer configuração feita. 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 forma objetiva e clara com a área de pré visualização e a árvore dos objetos auxiliando no acesso a eles.

Ao criar um jogo com a antiga ferramenta, foi visto que...

5 Conclusão e trabalhos futuros

Neste trabalho, apresentou-se uma melhoria significativa 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.

Para trabalhos futuros seria interessante disponibilizar mais ações na área de pré visualização que, atualmente, conta apenas com a função de mover os objetos para novas posições na tela, sendo interessante incluir outras propriedades. Além disso, 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.

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 2 vezes nas páginas 13 e 14.

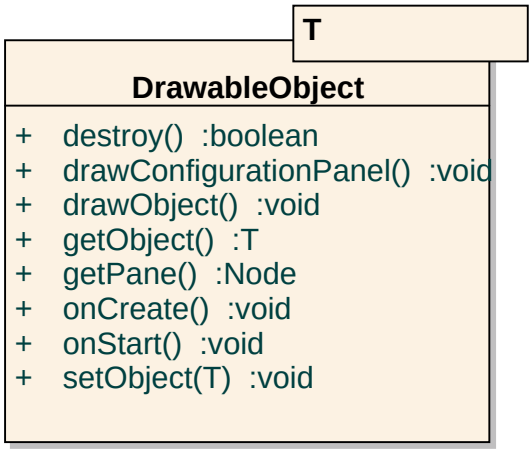
CGT. *Ceará Game Tools: Sobre o Projeto*. 2015. Disponível em: <http://www.cgt.ifce.edu.br/sobre.php>. Acesso em: 15.11.2015. Citado na página 13.

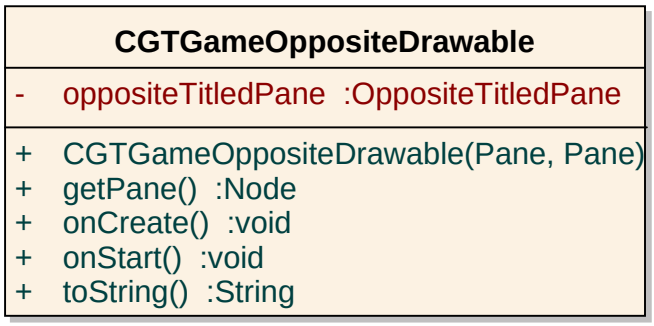
CGT. *Repositório do código fonte do projeto Ceará Game Tools hospedado na plataforma Github*. 2015. Disponível em: <https://github.com/hexat/projetocgt>. Acesso em: 9.12.2015. Citado na página 45.

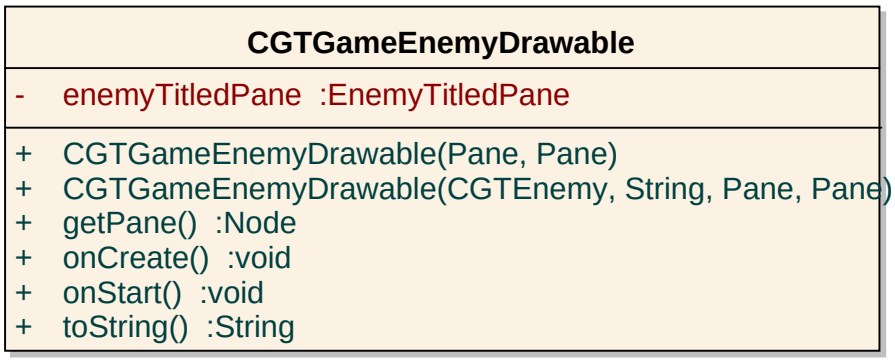
Anexos

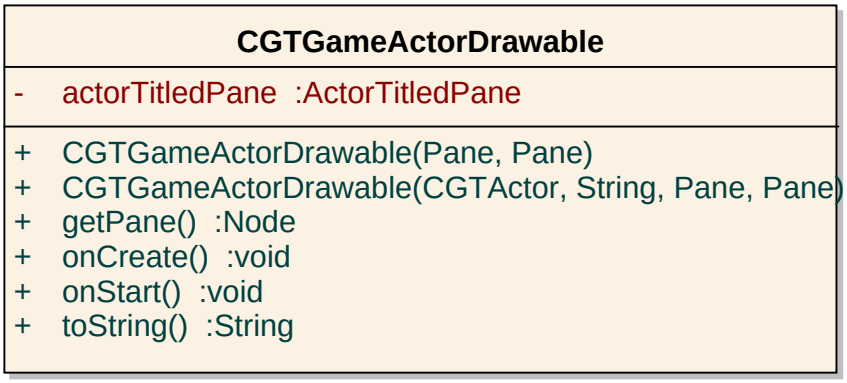
ANEXO A – Diagramas de classes

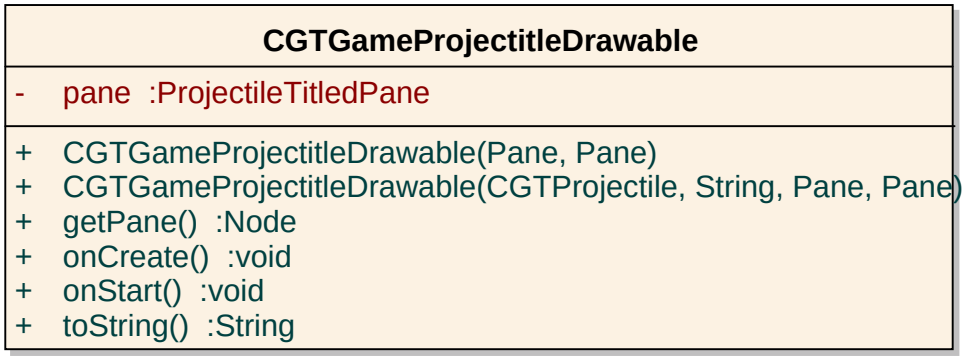
Dessa página em diante, pode-se ver o diagrama de classes do módulo que foi e implementado juntamente com esse trabalho, os diagramas são das classes presentes na `ferramenta` contidas no projeto CGT no pacote `br.edu.ifce.cgt.application.vo` e outras classes que precisaram ser adaptadas para a nova 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.

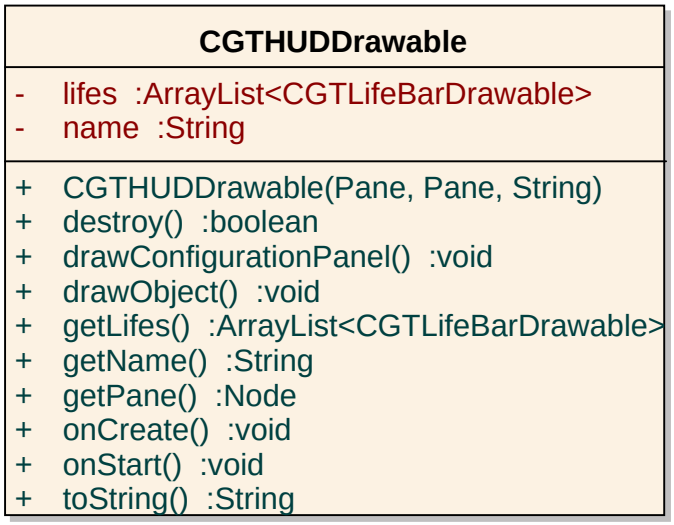

T
DrawableObject
+ destroy() :boolean
+ drawConfigurationPanel() :void
+ drawObject() :void
+ getObject() :T
+ getPane() :Node
+ onCreate() :void
+ onStart() :void
+ setObject(T) :void

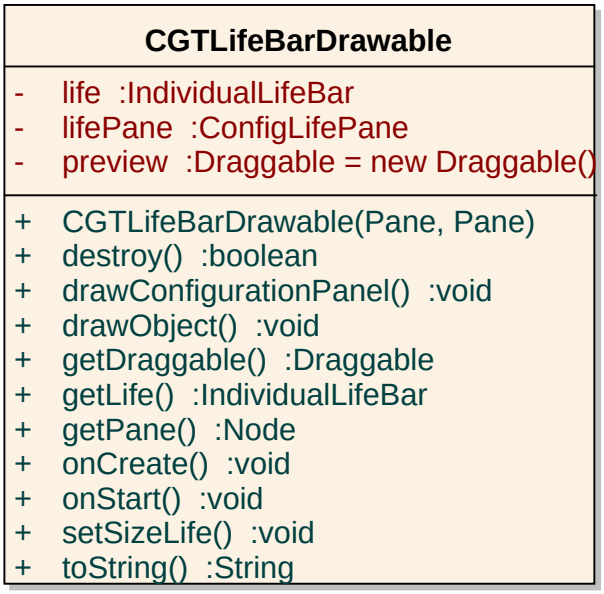

CGTGameOppositeDrawable
- oppositeTitledPane :OppositeTitledPane
+ CGTGameOppositeDrawable(Pane, Pane)
+ getPane() :Node
+ onCreate() :void
+ onStart() :void
+ toString() :String

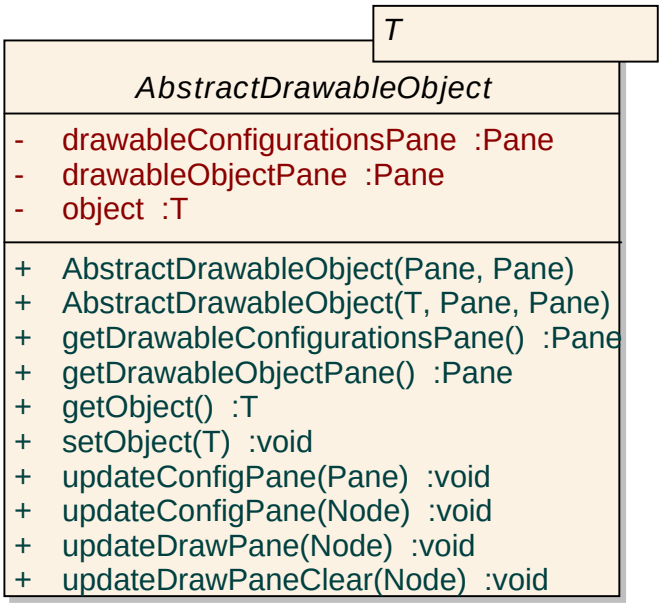

CGTGameEnemyDrawable
- enemyTitledPane :EnemyTitledPane
+ CGTGameEnemyDrawable(Pane, Pane)
+ CGTGameEnemyDrawable(CGTEnemy, String, Pane, Pane)
+ getPane() :Node
+ onCreate() :void
+ onStart() :void
+ toString() :String

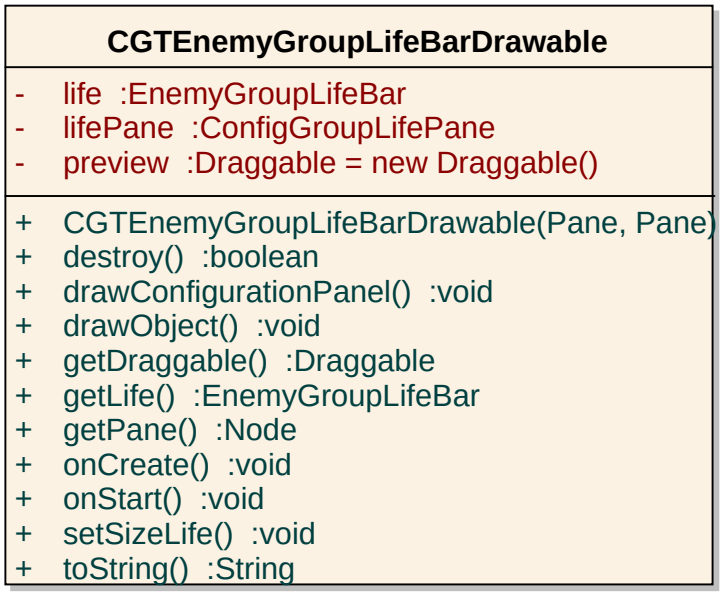

CGTGameActorDrawable
- actorTitledPane :ActorTitledPane
+ CGTGameActorDrawable(Pane, Pane)
+ CGTGameActorDrawable(CGTActor, String, Pane, Pane)
+ getPane() :Node
+ onCreate() :void
+ onStart() :void
+ toString() :String

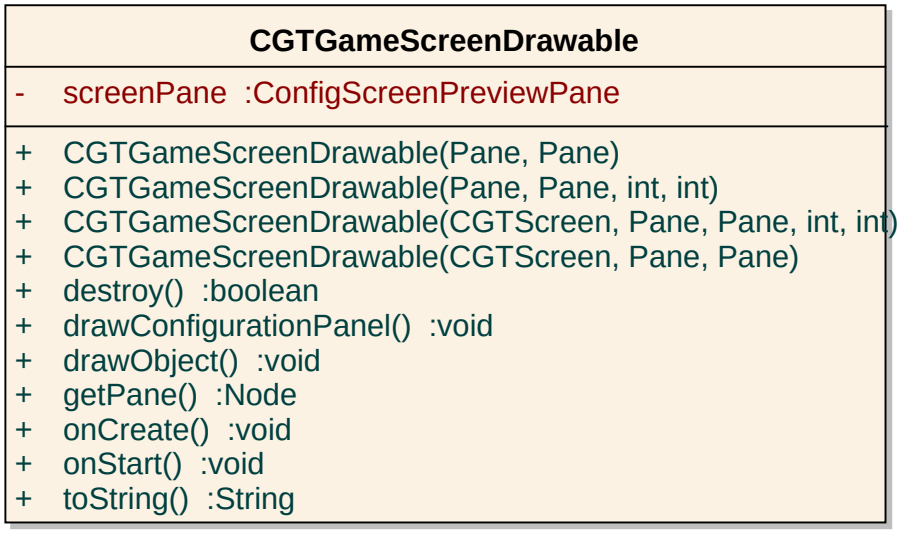

CGTGameProjectitleDrawable
- pane :ProjectileTitledPane
+ CGTGameProjectitleDrawable(Pane, Pane)
+ CGTGameProjectitleDrawable(CGTProjectile, String, Pane, Pane)
+ getPane() :Node
+ onCreate() :void
+ onStart() :void
+ toString() :String

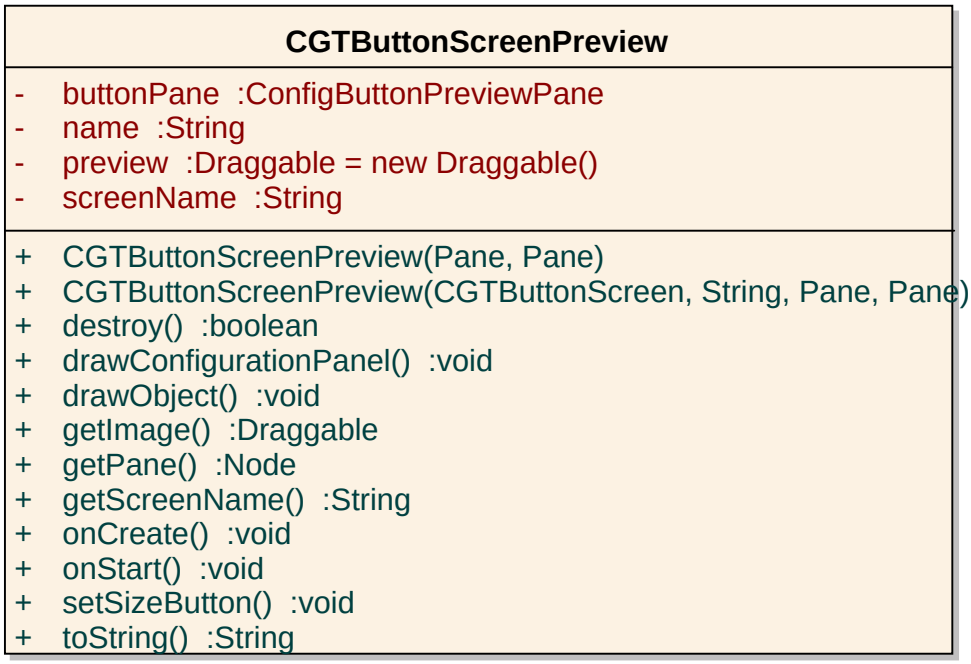

CGTHUDDrawable
- lifes :ArrayList<CGTLifeBarDrawable>
- name :String
+ CGTHUDDrawable(Pane, Pane, String)
+ destroy() :boolean
+ drawConfigurationPanel() :void
+ drawObject() :void
+ getLifes() :ArrayList<CGTLifeBarDrawable>
+ getName() :String
+ getPane() :Node
+ onCreate() :void
+ onStart() :void
+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

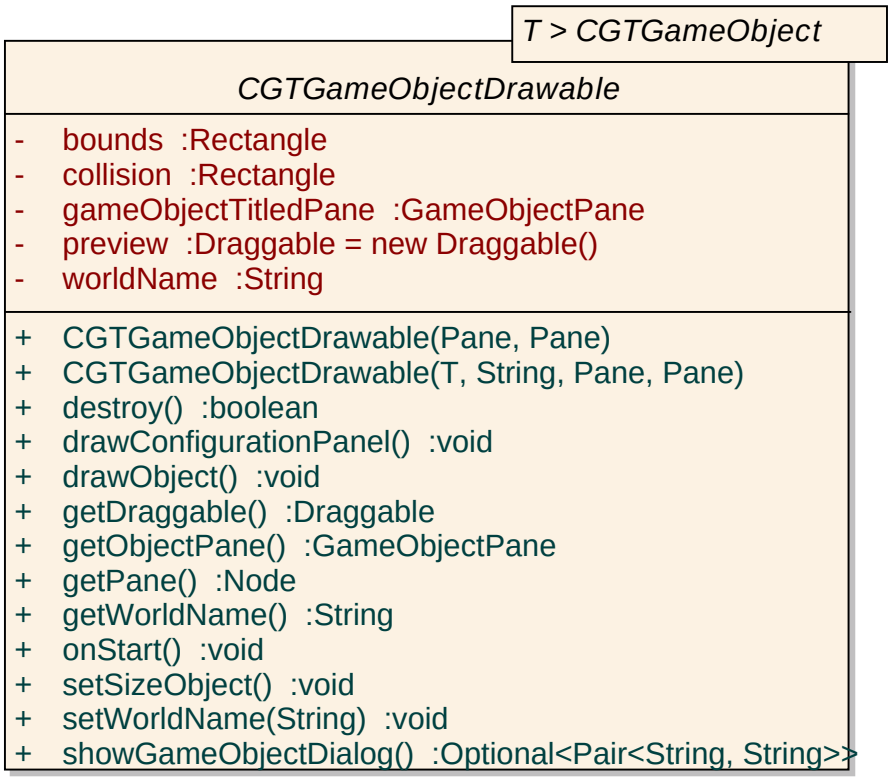

CGTLifeBarDrawable
- life :IndividualLifeBar
- lifePane :ConfigLifePane
- preview :Draggable = new Draggable()
+ CGTLifeBarDrawable(Pane, Pane)
+ destroy() :boolean
+ drawConfigurationPanel() :void
+ drawObject() :void
+ getDraggable() :Draggable
+ getLife() :IndividualLifeBar
+ getPane() :Node
+ onCreate() :void
+ onStart() :void
+ setSizeLife() :void
+ toString() :String


T
AbstractDrawableObject
- drawableConfigurationsPane :Pane
- drawableObjectPane :Pane
- object :T
+ AbstractDrawableObject(Pane, Pane)
+ AbstractDrawableObject(T, Pane, Pane)
+ getDrawableConfigurationsPane() :Pane
+ getDrawableObjectPane() :Pane
+ getObject() :T
+ setObject(T) :void
+ updateConfigPane(Pane) :void
+ updateConfigPane(Node) :void
+ updateDrawPane(Node) :void
+ updateDrawPaneClear(Node) :void


CGTEnemyGroupLifeBarDrawable
- life :EnemyGroupLifeBar
- lifePane :ConfigGroupLifePane
- preview :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
- screenPane :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


T > CGTGameObject
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>>


BorderPane
PreviewPane
+ DESKTOP_JAR_PATH :String = "desktop/deskto... {readOnly}
+ DESKTOP_ZIP_PATH :String = "desktop/deskto... {readOnly}
+ drawableConfigurationsPane :Pane
+ drawableObjectPane :Pane
- openRecentMenu :Menu
- rootItem :CGTProjectDrawable
- running :boolean
- tree :TreeView<DrawableObject>
+ about() :void
+ addBonus() :void
+ addButtonScreen() :void
+ addEnemy() :void
+ addEnemyLifeBar() :void
+ addGearInformation() :void
+ addObjectLifeBar() :void
+ addOpponent() :void
+ addProjectile() :void
+ addScreen() :void
+ addSpriteSheet() :void
+ addWorld() :void
+ beforeClosing() :void
+ closeProject() :void
- copyDesktopFiles() :void
+ editSpriteSheet() :void
+ exit() :void
+ exportProject() :void
- getActorWorldNode(String) :TreeItem<DrawableObject>
- getHUDNode(String) :TreeItem<DrawableObject>
- getScreenNode(String) :TreeItem<DrawableObject>
- getWorldNode(String) :TreeItem<DrawableObject>
- isWin() :boolean
- localDefaultDirectory() :String
+ newProject() :void
- open(File) :void
+ openProject() :void
+ PreviewPane()
- runDesktop() :void
+ runProject() :void
+ saveProject() :void
+ saveProjectAs() :void
- setupTree() :void
- showValidateDialog(List<CGTError>) :void
- updateRecent() :void
- updateTree() :void