Documentação

3R: Ar

Felipe Sampaio de Souza
Gabriel de Paiva Oliveira
Miguel de Souza Tosta
Pedro Henrique Migliatti

São Carlos - 2016

Sumário

1. Autores

a. Identificação

- Felipe Sampaio de Souza
  - RA: 619523
  - E-mail: sampaio.motion@gmail.com

- Gabriel de Paiva Oliveira
  - RA: 619566;
  - E-mail: gabriel.paiv.oliv@hotmail.com

- Miguel de Souza Tosta
  - RA: 619698
  - E-mail: miguel_st02@hotmail.com

- Pedro Henrique Migliatti
  - RA: 619744
  - E-mail: pedro.migliatti@gmail.com

b. Participação

Os quatro integrantes do grupo se dividiram em duplas para poder desenvolver os dois jogos em paralelo. Essa divisão se aplica apenas a implementação da lógica dos jogos pois, além das duplas se supervisionarem e ajudarem, o conceito e ideia dos jogos foram discutidos e trabalhados em equipe.

A dupla responsável pela implementação descrita nesta documentação foi: Felipe Sampaio de Souza e Pedro Henrique Migliatti.

Além da implementação dos jogos o grupo também trabalhou na pesquisa das estrutura de dados e da biblioteca gráfica em conjunto, de forma que ambos os jogos se beneficiassem com as pesquisas e descobertas.

2. Jogabilidade e Funcionamento do Jogo

a. Introdução

Seguindo a ideia do primeiro jogo da disciplina, este jogo tem como tema um elemento da natureza, neste caso o ar e, também, tem a intenção de proporcionar para o usuário algo mais que apenas entretenimento, sendo educacional.

Nesse jogo o usuário precisa controlar um foguete espacial com o objetivo de chegar ao espaço. Durante o caminho o jogador passará por todas as camadas da atmosfera terrestre: Troposfera, Estratosfera, Mesosfera, Termosfera e Exosfera.

b. Controles

No jogo os comandos são via teclado e mouse, sendo a personagem principal (Nave) controlada pelas setas e pela barra de espaço do teclado e as telas que aparecem durante o jogo são controladas pelo mouse através dos botões existente nelas (o mouse interage com o jogo apenas através do botão esquerdo).

Abaixo encontramos uma tabela da teclas que interagem com o jogo.

Tecla	Função
Left ←	Gira o foguete para o lado esquerdo.
Right →	Gira o foguete para o lado direito.
Espaço	Quando o foguete está parada: Inicia a decolagem da foguete. Quando o foguete está em movimento: Reinicia o jogo.
Esc	Fecha o jogo.

c. Dinâmica

Inicialmente o foguete do jogo está parado na Terra, seu objetivo é decolar e atingir a camada mais alta da atmosfera terrestre para ir ao espaço. Durante a sua viagem até a Exosfera o jogador encontra vários obstáculos que dificultarão sua chegada ao espaço, como por exemplo a poluição encontrada na Troposfera e os cometas na Mesosfera. Além dos obstáculos o jogador se deparará com ícones de aprimoramento que irão ajudar na sua jornada. Esses icones são coletados quando colidem com a nave e assim que o jogador reinicia o jogo ou perde (suas vidas acabam ou a bateria esvazia) esses aprimoramentos são passados para sua árvore de aprimoramento e esses são ativos na próxima rodada.

Os obstáculos que o jogador poderá encontrar durante sua jornada são:

- Troposfera: Poluição (Nuvens negras)
- Estratosfera: Balões
- Mesosfera: Cometas
- Entre a Mesosfera e a Termosfera: Camada de Ozônio.
- Termosfera: Satélites

Todos os obstáculos retiram do jogador uma vida (representada pelas chaves inglesas no canto superior direito da tela) quando colidem com o foguete, menos a camada de Ozônio que retira do jogador duas vidas em uma única colisão. Cada tipo de obstáculo tem um número fixo de instâncias que serão geradas em cada rodada (com exceção do cometa que

tem esse valor alterado quando um determinado aprimoramento é desbloqueado), porém a posição na camada referente ao tipo de obstáculo é gerada de maneira aleatória sempre que o jogador inicia uma partida.

Os aprimoramentos que podem ser encontrados no cenário estão distribuídos de forma que cada um tenha uma camada especifica onde pode ser encontrado, porém sua posição é gerada de maneira aleatória, para aumentar a dificuldade do jogo e deixar mais divertida cada rodada. O jogador pode coletar um aprimoramento por rodada e essa coleta é feita através da colisão da nave com o ícone do aprimoramento que estará flutuando pelo cenário.

Quando o jogador perde ou encerra a rodada, caso ele tenha colidido com algum ícone de aprimoramento, o aprimoramento é habilitado na árvore de aprimoramentos do jogador e quando o jogador iniciar a próxima rodada esse aprimoramento já estará valendo.

Os aprimoramentos existentes no jogo e suas respectivas posições na árvore de aprimoramentos são:

- Nó Raiz: Já vem habilitado e não possuí nada de diferente;
- Nó da Direito: Aumenta o número de vidas do jogador e diminui o atrito com os obstáculos;
- Nó da Esquerdo: Retira a poluição da Troposfera, as nuvens ficam brancas e não causam mais dano ao foguete;
- Nó Direito da Subárvore da Direita: Aumenta a duração do combustível do foguete;
- Nó Esquerdo da Subárvore da Direita: Aumenta a velocidade do foguete;
- Nó Direito da Subárvore da Esquerda: Diminui a quantidade de cometas na Mesosfera;
- Nó Esquerdo da Subárvore da Esquerda: Restaura a camada de ozônio, assim ela não causará mais dano ao foguete.

Os ícones de aprimoramento só apareceram no cenário para o jogador caso o Nó acima desse aprimoramento na árvores esteja ativo.

Quando o foguete está parado o jogador inicia sua decolagem apertando a tecla “Espaço” no teclado, assim o foguete começa a se movimentar para cima. Durante a sua viagem o jogador deve usar das setas do teclado (Esquerda e Direita) para movimentar o foguete, desviando de obstáculos e adquirindo aprimoramentos.

Sempre que o jogador reinicia o jogo (apertando “Espaço” quando o foguete está em movimento) ou perde a rodada (quando suas vidas acabam ou o combustível termina) é exibida uma tela para o jogador com informações sobre a camada atmosférica onde ele parou. Quando o jogador atinge a Exosfera seu foguete vai para o espaço e é exibida a tela de término do jogo.

Abaixo estão algumas imagens que ajudam a entender a dinâmica do jogo e completam o texto descritivo acima.

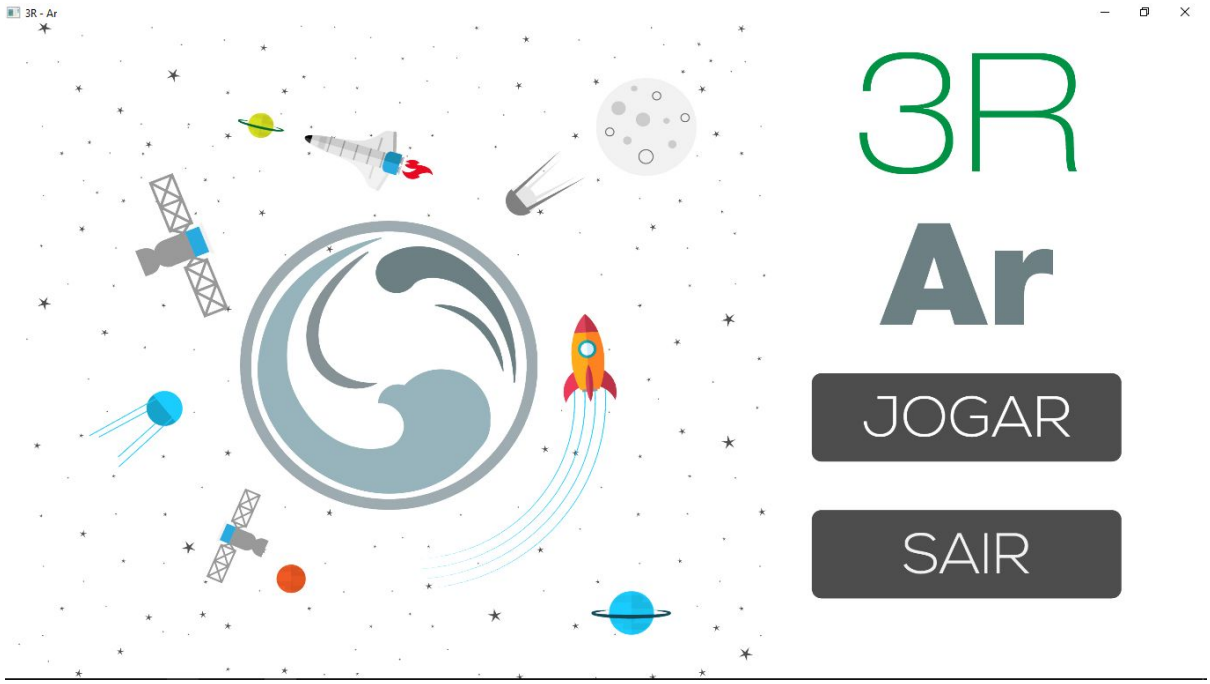

Imagem 1. Tela inicial do Jogo

Imagem 2. Visão do inicio da rodada (Foguete parado)

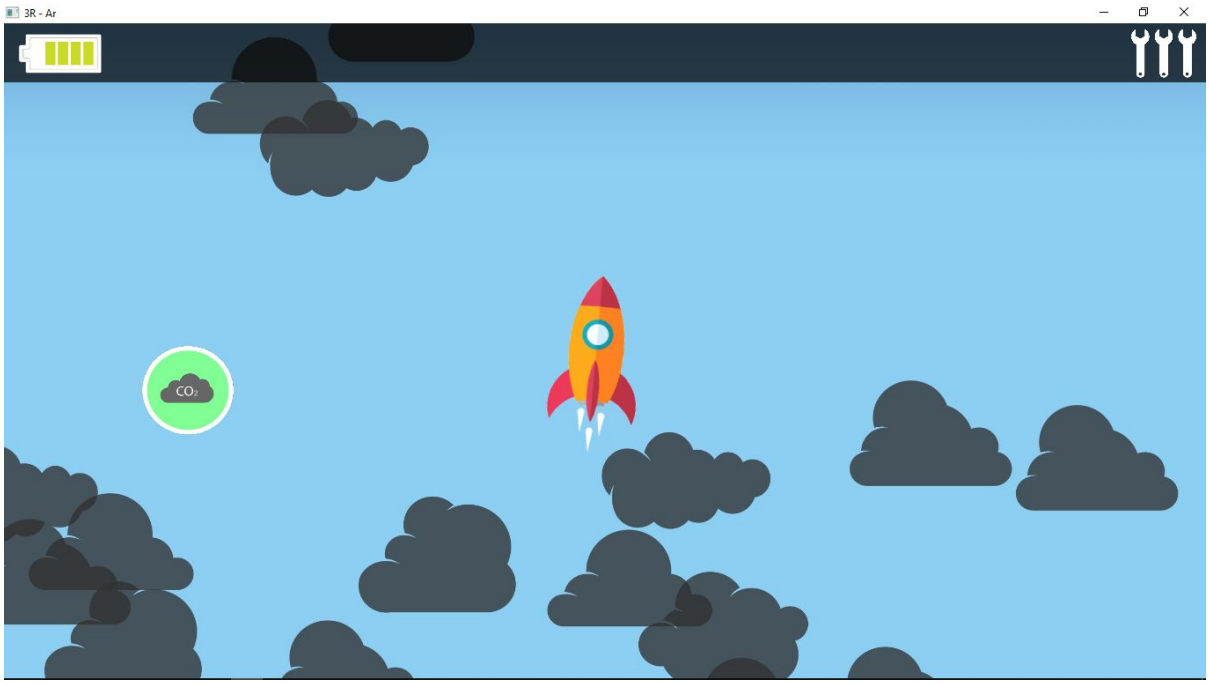

Imagem 3. Foguete na Troposfera com nuvens poluídas e um ícone de aprimoramento

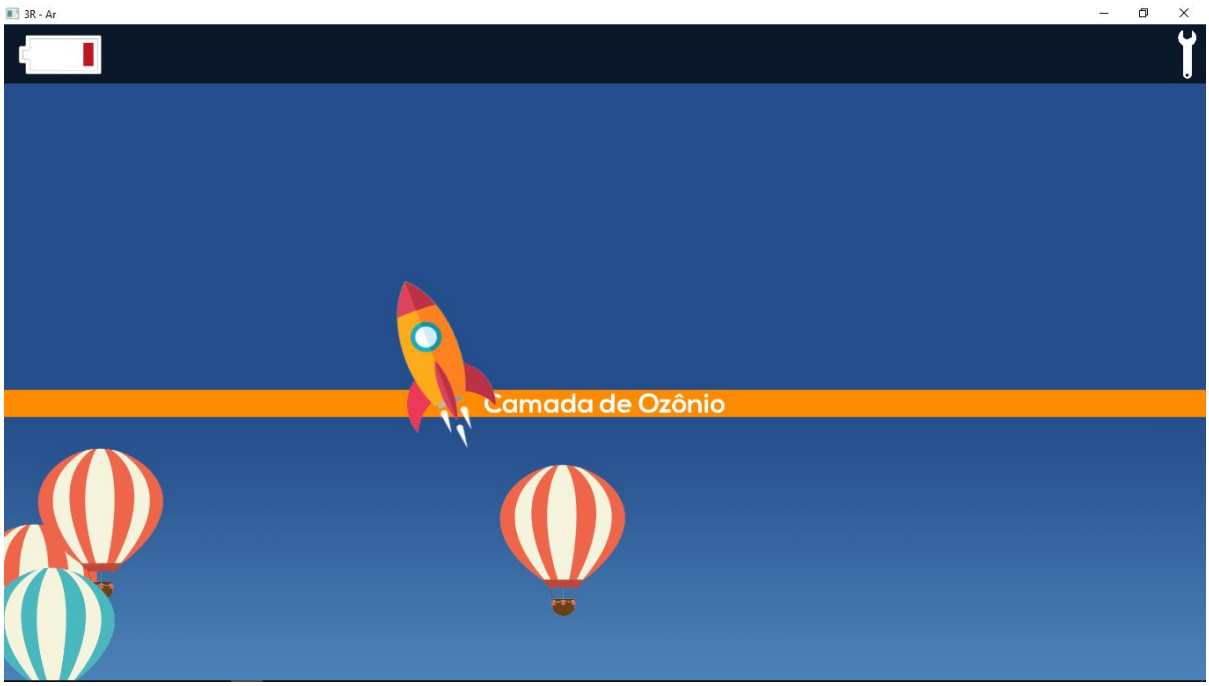

Imagem 4. Foguete passando pela camada de ozônio ainda danificada pela poluição

Imagem 5. Tela do jogo pausado quando o foguete parou na Troposfera

Imagem 6. Tela final do jogo

d. Implementação e ferramentas

Para a execução da parte gráfica do jogo, foi utilizada a biblioteca *SFML (Simple Fast Media Library)*, uma biblioteca multiplataforma focada na implementação em C++, por esse motivo, a linguagem utilizada para implementação do projeto foi C++.

Observação: Para que o executável funcione, ele necessita estar na mesma pasta que os arquivos ".dll" da biblioteca *SFML*.

3. Estrutura: Árvore

a. Introdução

A estrutura de dados utilizada no jogo foi a árvore, porém o grupo não se prendeu em implementar a estrutura apenas para ser utilizada na dinâmica do jogo, mas também na forma de gerenciamento dos gráficos, o que facilitou a implementação do jogo e fez os integrantes do grupo terem um entendimento melhor da estrutura Árvore, de funções recursivas e do uso de polimorfismo na linguagem C++.

b. Implementação da Estrutura

Podemos dividir a implementação da estrutura em duas partes: a parte que trata da dinâmica do jogo, que é uma árvore binária simples que está implementada na classe Skill do jogo. Nessa classe temos um atributo do tipo "SkillsNode*", ou seja, um ponteiro para a classe SkillsNode, é nessa classe que está implementados o Nó da árvore binária.

SkillsNode - Essa classe além de conter um "Enum" com todos os aprimoramentos que podem ser implementados no jogo, também contém dois ponteiros do tipo SkillsNode*, que são os ponteiros para os filhos do nó.

```
14  class SkillsNode: public NodeCena
15  {
16      public:
17          enum Aprimoramento
18          {
19              NAVE,
20              SHIELD,
21              CO2,
22              SPEED,
23              GAS,
24              SPRAY,
25              COMETA,
26              AprimoramentoCount
27          };
28                          SkillsNode(const sf::Texture&, const sf::Texture&, const sf::Texture&, const sf::Texture&, int);
29                          ~SkillsNode();
30          sf::FloatRect   getBoundingRect() const;
31          void            desenhaAtual(sf::RenderTarget&, sf::RenderStates) const;
32
33          int             getAltura();
34          bool            mouseIntersects(sf::RenderWindow& window);
35          bool            isPressed(sf::RenderWindow& window);
36          bool            isSelected(sf::RenderWindow& window);
37
38
39          SkillsNode*     Dir;
40          SkillsNode*     Esq;
41
42          bool            status, desc;
43          std::array<sf::Sprite,5> simbolo; // status= false -> simbolo[0], status= true -> simbolo[1]
44          void            draw(sf::RenderTarget& target, sf::RenderStates states) const;
45          int             aprimoramento;
46  };
```

A outra parte da implementação da estrutura árvore é referente a parte gráfica do jogo e está dividida principalmente em duas classes: NodeCena e SpriteNode, as quais as implementações podem ser verificadas abaixo. Essa tipo de implementação ajuda os desenvolvedores, pois utilizando a estrutura de árvore podemos fazer elementos do jogo que serão desenhados na tela serem filhos um dos outros caso haja dependência e a utilização de algoritmos recursivos também ajudam na implementação do jogo.

NodeCena

```
11
12  class NodeCena : public sf::Transformable, public sf::Drawable, private sf::NonCopyable
13  {
14  public:
15      NodeCena();
16
17      void                    insereFilho(NodeCena* filho);
18      void                    esvaziaFilhos();
19      std::vector<NodeCena*>  getFilhos();
20
21      void                    atualiza(sf::Time delta);
22
23      sf::Vector2f            getWorldPosition() const;
24      sf::Transform           getWorldTransform() const;
25      virtual sf::FloatRect   getBoundingRect() const;
26      virtual void            setOriginCenter();
27
28      void                    desenhaFilhos(sf::RenderTarget& target, sf::RenderStates states) const;
29
30  private:
31      virtual void           atualizaAtual(sf::Time delta);
32      void                   atualizaFihos(sf::Time delta);
33
34      virtual void           draw(sf::RenderTarget& target, sf::RenderStates states) const;
35      virtual void           desenhaAtual(sf::RenderTarget& target, sf::RenderStates states) const;
36
37  protected:
38      std::vector<NodeCena*>  filhos;
39      NodeCena*       pai;
40  };
```

SpriteNode

```
9   class SpriteNode : public NodeCena
10  {
11  public:
12                              SpriteNode(const sf::Texture& texture);
13                              SpriteNode(const sf::Texture& texture, const sf::IntRect& textureRect);
14      void                    setOriginCenter();
15      virtual sf::FloatRect   getBoundingRect() const;
16      sf::Sprite              getSprite() const;
17      void                    setTexture(const sf::Texture& texture);
18  private:
19
20      virtual void            desenhaAtual(sf::RenderTarget& target, sf::RenderStates states) const;
21
22  private:
23      sf::Sprite              sprite;
24  };
```

Diagrama da árvore de cenas do jogo - Esse diagrama representa a árvore utilizada para renderizar (exibir na tela) os objetos e demais imagens do jogo. Aqui podemos verificar que cada cena é dividida em vários Nós, cada um representando uma camada dessa cena, assim é possível fazer com que elementos se sobreponham da maneira adequada.

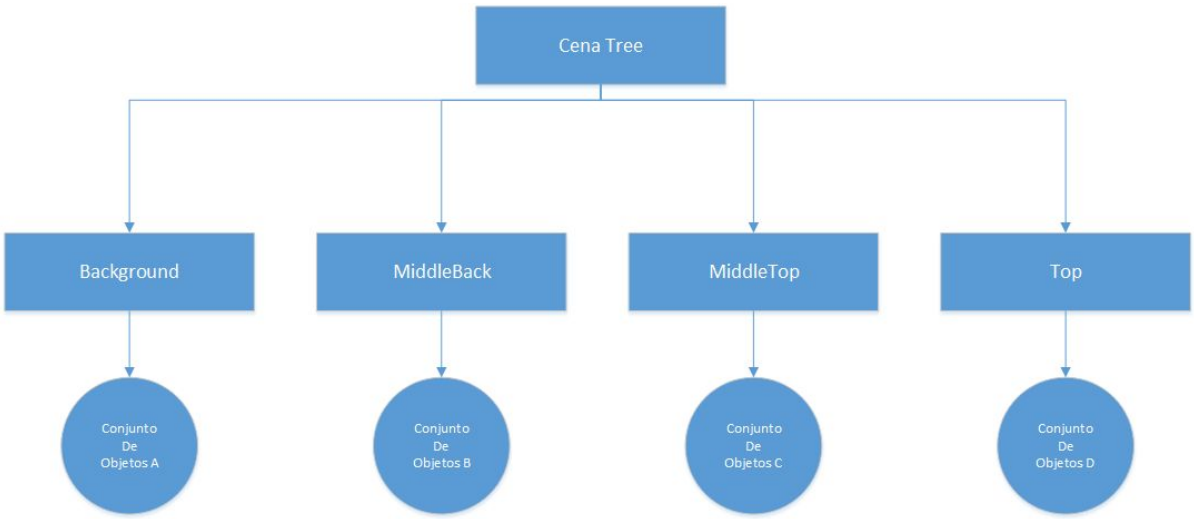

Imagem 7. Diagrama de classes.

4. Diagrama da Arquitetura do Jogo

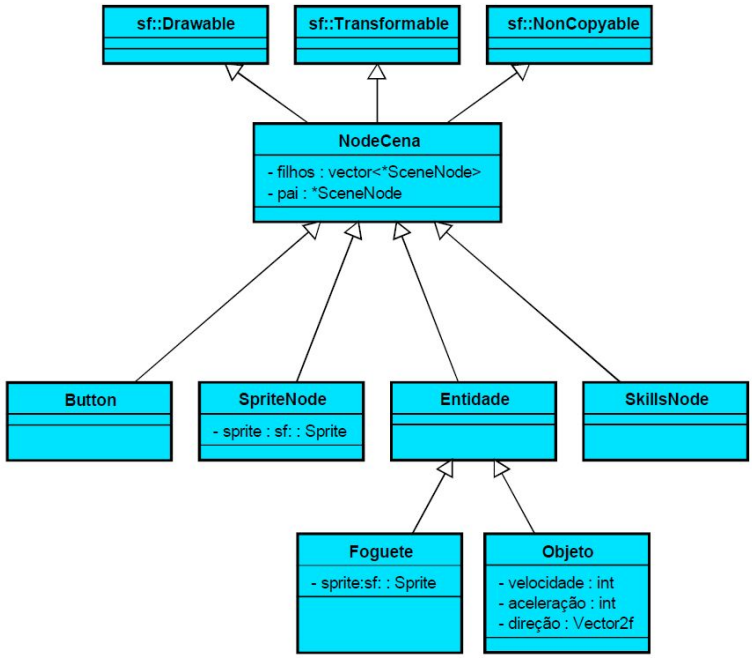

Imagem 8. Diagrama das principais classes.

5. Conclusão

Assim como no primeiro jogo desenvolvido pelo grupo, este também trouxe grandes desafios, porém desafios diferentes dos anteriores. Dessa vez o grupo se dividiu de maneira mais eficiente, procurando evoluir e melhorar a metodologia de desenvolvimento do projeto.

A equipe também encontrou o desafio de manter as ideias do jogo anterior, uma meta decidida pelo grupo durante o desenvolvimento do primeiro jogo, isso fez com que o grupo trabalhasse muito na ideia e na jogabilidade do programa, passando por várias mudanças até mesmo durante a implementação do mesmo.

O desenvolvimento deste jogo possibilitou aos integrantes do grupo se aprofundar nos recursos da biblioteca gráfica, na estrutura de dados e na linguagem de programação utilizadas.

6. Referências

- Site da Biblioteca Gráfica Utilizada (SFML): http://www.sfml-dev.org/
- Livro: SFML Game Development (Haller, J.; Hansson, H. V.; Moreira, A.)
- Tutorial SFML: http://www.sfml-dev.org/tutorials/2.3/
- Softwares: Code::Blocks 16.01, pacote Adobe CS6
- Imagens: www.freepik.com/
- Áudios: www.freesound.org
- Música: www.incompetech.com/music/