Peitche!

Generated by Doxygen 1.14.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

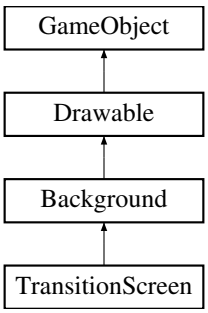
Class Documentation

4.1 Background Class Reference

Classe que representa um plano de fundo do jogo.

`#include <passive.hpp>`

Inheritance diagram for Background:



Public Member Functions

- Background (const char *dir, const Point &pos, float w, float h, float speedX)

  Construtor do plano de fundo.
- virtual ~**Background** ()

  Destrutor virtual do plano de fundo.
- void updateSpeed () override

  Atualiza a velocidade do plano de fundo.
- bool updatePosition () override

  Atualiza a posição do plano de fundo.
- void draw () override

  Desenha o plano de fundo na tela.

Public Member Functions inherited from Drawable

- void addSpeedVector (const Point &spd)

  Adiciona um vetor à velocidade atual.
- void addSpeedX (float x)

  Adiciona um valor à velocidade X.
- void addSpeedY (float y)

  Adiciona um valor à velocidade Y.
- Point getSpeed ()

  Retorna o vetor de velocidade.
- float getSpeedX ()

  Retorna a velocidade no eixo X.
- float getSpeedY ()

  Retorna a velocidade no eixo Y.

Public Member Functions inherited from GameObject

- Point getPos ()

  Retorna a posição do objeto.
- float getPosX ()

  Retorna a coordenada X do objeto.
- float getPosY ()

  Retorna a coordenada Y do objeto.

Protected Attributes

- ALLEGRO_BITMAP * **image** = nullptr

  Bitmap do plano de fundo.
- float **width**

  Largura do plano de fundo.
- float **height**

  Altura do plano de fundo.

Friends

- class **BackgroundHandler**

Additional Inherited Members

Protected Member Functions inherited from Drawable

- Drawable (const Point &pos, const Point &spd)

  Construtor protegido com posição e velocidade.
- Drawable (const Point &pos)

  Construtor protegido apenas com posição.
- void setSpeedX (float x)

  Define a velocidade no eixo X.
- void setSpeedY (float y)

  Define a velocidade no eixo Y.
- void setSpeed (const Point &pos)

  Define o vetor de velocidade.

Protected Member Functions inherited from GameObject

- GameObject (const Point &position)

  Construtor protegido com posição.
- GameObject (float pX, float pY)

  Construtor protegido com coordenadas.
- void setPosX (float x)

  Define a coordenada X do objeto.
- void setPosY (float y)

  Define a coordenada Y do objeto.
- void setPos (const Point &p)

  Define a posição do objeto.

4.1.1 Detailed Description

Classe que representa um plano de fundo do jogo.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Background()

```
Background::Background (
            const char * dir,
            const Point & pos,
            float w,
            float h,
            float speedX)
```

Construtor do plano de fundo.

Parameters

dir	Caminho da imagem.
pos	Posição inicial.
w	Largura.
h	Altura.
speedX	Velocidade no eixo X.

4.1.3 Member Function Documentation

4.1.3.1 draw()

```
void Background::draw ()  [override], [virtual]
```

Desenha o plano de fundo na tela.

Implements Drawable.

Reimplemented in TransitionScreen.

4.1.3.2 updatePosition()

```
bool Background::updatePosition ()  [override], [virtual]
```

Atualiza a posição do plano de fundo.

Returns

true se a posição foi atualizada.

Implements GameObject.

Reimplemented in TransitionScreen.

4.1.3.3 updateSpeed()

```
void Background::updateSpeed ()  [override], [virtual]
```

Atualiza a velocidade do plano de fundo.

Implements Drawable.

Reimplemented in TransitionScreen.

The documentation for this class was generated from the following files:

- include/passive.hpp
- src/passive.cpp

4.2 BackgroundHandler Class Reference

Classe responsável por gerenciar múltiplos planos de fundo.

```
#include <passive.hpp>
```

Public Member Functions

- BackgroundHandler (const char *dir, float w, float h, float speedX, float screenW, float screenH)
  Construtor do gerenciador de planos de fundo.
- ~**BackgroundHandler** ()
  Destrutor do gerenciador de planos de fundo.
- void **drawBackground** ()
  Desenha todos os planos de fundo.
- void **updateBackgroundPosition** ()
  Atualiza a posição dos planos de fundo.

4.2.1 Detailed Description

Classe responsável por gerenciar múltiplos planos de fundo.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 BackgroundHandler()

```
BackgroundHandler::BackgroundHandler (
            const char * dir,
            float w,
            float h,
            float speedX,
            float screenW,
            float screenH)
```

Construtor do gerenciador de planos de fundo.

Parameters

dir	Caminho da imagem.
w	Largura do plano de fundo.
h	Altura do plano de fundo.
speedX	Velocidade no eixo X.
screenW	Largura da tela.
screenH	Altura da tela.

The documentation for this class was generated from the following files:

- include/passive.hpp
- src/passive.cpp

4.3 Base Class Reference

Classe que representa a base de dados de perfis.

```
#include <base.hpp>
```

Public Member Functions

- Base (const Base &)
  Construtor de cópia.
- Base (const vector< Profile * > &)
  Construtor a partir de um vetor de perfis.
- **Base** ()
  Construtor padrão.
- Base (string)
  Construtor que carrega base de um arquivo.
- ~**Base** ()
  Destrutor.
- Base operator= (const Base &)
  Operador de atribuição.
- bool inBase (string)

Verifica se um nickname está na base.

- bool inBase (Profile)

  Verifica se um perfil está na base.
- bool updateProfiles (Profile)

  Atualiza ou adiciona um perfil na base.
- bool removeProfile (string)

  Remove um perfil da base pelo nickname.
- vector< Profile * > getBestProfiles ()

  Retorna os melhores perfis da base.
- void saveBase (string)

  Salva a base em um arquivo.
- void **display** ()

  Exibe todos os perfis da base.

4.3.1 Detailed Description

Classe que representa a base de dados de perfis.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Base() [1/3]

```
Base::Base (
            const Base & otherBase)
```

Construtor de cópia.

Parameters

other	Outro objeto Base.

4.3.2.2 Base() [2/3]

```
Base::Base (
            const vector< Profile * > & otherBaseVector)
```

Construtor a partir de um vetor de perfis.

Parameters

profiles	Vetor de ponteiros para perfis.

4.3.2.3 Base() [3/3]

```
Base::Base (
            string path)
```

Construtor que carrega base de um arquivo.

Parameters

filename	Nome do arquivo.

4.3.3 Member Function Documentation

4.3.3.1 getBestProfiles()

```
vector< Profile * > Base::getBestProfiles ()
```

Retorna os melhores perfis da base.

Returns

vector<Profile *> Vetor dos melhores perfis.

4.3.3.2 inBase() [1/2]

```
bool Base::inBase (
            Profile otherProfile)
```

Verifica se um perfil está na base.

Parameters

profile	Perfil a ser verificado.

Returns

true se está na base.

4.3.3.3 inBase() [2/2]

```
bool Base::inBase (
            string nickname)
```

Verifica se um nickname está na base.

Parameters

nickname	Nickname a ser verificado.

Returns

true se está na base.

4.3.3.4 operator=()

```
Base Base::operator= (
            const Base & otherBase)
```

Operador de atribuição.

Parameters

other	Outro objeto Base.

Returns

Base& Referência para este objeto.

4.3.3.5 removeProfile()

```
bool Base::removeProfile (
            string nickname)
```

Remove um perfil da base pelo nickname.

Parameters

nickname	Nickname do perfil a ser removido.

Returns

true se removido com sucesso.

4.3.3.6 saveBase()

```
void Base::saveBase (
            string path)
```

Salva a base em um arquivo.

Parameters

filename	Nome do arquivo.

4.3.3.7 updateProfiles()

```
bool Base::updateProfiles (
            Profile newProfile)
```

Atualiza ou adiciona um perfil na base.

Parameters

profile	Perfil a ser atualizado/adicionado.

Returns

true se atualizado/adicionado com sucesso.

The documentation for this class was generated from the following files:

- include/base.hpp
- src/base.cpp

4.4 Color Struct Reference

Estrutura que representa uma cor RGB.

`#include <table.hpp>`

Public Member Functions

- **Color** ()
  Construtor padrão.
- Color (float r, float g, float b)
  Construtor com valores RGB.
- Color (const Color &other)
  Construtor de cópia.
- Color & operator= (const Color &other)
  Operador de atribuição.
- Color operator/ (float f) const
  Divide cor por escalar.
- void **display** ()
  Exibe a cor no console.

Public Attributes

- float **r**
- float **g**
- float **b**
  Componentes de cor (0 - 255)

4.4.1 Detailed Description

Estrutura que representa uma cor RGB.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Color() [1/2]

```
Color::Color (
            float r,
            float g,
            float b)
```

Construtor com valores RGB.

Parameters

r	Vermelho.
g	Verde.
b	Azul.

4.4.2.2 Color() [2/2]

```
Color::Color (
            const Color & other)
```

Construtor de cópia.

Parameters

other	Outra cor.

4.4.3 Member Function Documentation

4.4.3.1 operator/()

```
Color Color::operator/ (
            float f) const
```

Divide cor por escalar.

Parameters

f	Escalar.

Returns

Resultado da divisão.

4.4.3.2 operator=()

```
Color & Color::operator= (
            const Color & other)
```

Operador de atribuição.

Parameters

other	Outra cor.

Returns

Referência para esta cor.

The documentation for this struct was generated from the following files:

- include/table.hpp
- src/table.cpp

4.5 Cooldown Class Reference

Classe que gerencia cooldowns temporais.

```
#include <cooldown.hpp>
```

Public Member Functions

- Cooldown (float wuTime)

  Construtor que define o tempo de recarga em segundos.
- **Cooldown** ()

  Construtor padrão.
- void **updateCooldown** ()

  Atualiza o cooldown, reduzindo o tempo restante.
- void **restartCooldown** ()

  Reinicia o cooldown para o tempo de recarga.
- void **refreshCooldown** ()

  Define o cooldown como pronto imediatamente.
- bool isCooldownUp ()

  Verifica se o cooldown terminou.
- void setRechargeTime (float wuTime)

  Define um novo tempo de recarga (em segundos).
- void setUpdateFrequency (float f)

  Define um novo fator de velocidade de atualização.
- float getCurrentTimeLeft ()

  Retorna o tempo restante atual do cooldown.
- float getCurrentPorcentage ()

  Retorna a porcentagem do cooldown já decorrido.
- float getRechargeTime ()

  Retorna o tempo total de recarga.
- void **freezeCooldown** ()

  Congela o cooldown (pausa a contagem).
- void **unfreezeCooldown** ()

  Descongela o cooldown (retoma a contagem).

4.5.1 Detailed Description

Classe que gerencia cooldowns temporais.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Cooldown()

```
Cooldown::Cooldown (
            float wuTime)
```

Construtor que define o tempo de recarga em segundos.

Parameters

wuTime	Tempo de recarga em segundos.

4.5.3 Member Function Documentation

4.5.3.1 getCurrentPorcentage()

```
float Cooldown::getCurrentPorcentage ()
```

Retorna a porcentagem do cooldown já decorrido.

Returns

Porcentagem (0 a 1).

4.5.3.2 getCurrentTimeLeft()

```
float Cooldown::getCurrentTimeLeft ()
```

Retorna o tempo restante atual do cooldown.

Returns

Tempo restante.

4.5.3.3 getRechargeTime()

```
float Cooldown::getRechargeTime ()
```

Retorna o tempo total de recarga.

Returns

Tempo de recarga.

4.5.3.4 isCooldownUp()

```
bool Cooldown::isCooldownUp ()
```

Verifica se o cooldown terminou.

Returns

true se terminou, false caso contrário.

4.5.3.5 setRechargeTime()

```
void Cooldown::setRechargeTime (
            float wuTime)
```

Define um novo tempo de recarga (em segundos).

Parameters

wuTime	Novo tempo de recarga.

4.5.3.6 setUpdateFrequency()

```
void Cooldown::setUpdateFrequency (
            float f)
```

Define um novo fator de velocidade de atualização.

Parameters

f	Novo fator de atualização.

The documentation for this class was generated from the following files:

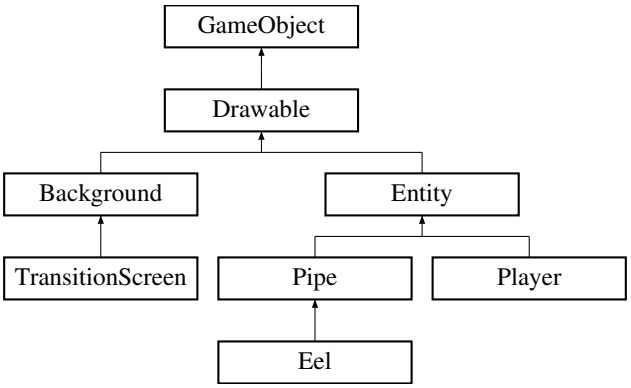
- include/cooldown.hpp
- src/cooldown.cpp

4.6 Drawable Class Reference

Classe base para todos os objetos desenháveis e que podem se mover.

```
#include <game_object.hpp>
```

Inheritance diagram for Drawable:



Public Member Functions

- virtual void draw ()=0

  Desenha o objeto na tela.
- virtual void updateSpeed ()=0

  Atualiza a velocidade do objeto.
- void addSpeedVector (const Point &spd)

  Adiciona um vetor à velocidade atual.
- void addSpeedX (float x)

  Adiciona um valor à velocidade X.
- void addSpeedY (float y)

  Adiciona um valor à velocidade Y.
- Point getSpeed ()

  Retorna o vetor de velocidade.
- float getSpeedX ()

  Retorna a velocidade no eixo X.
- float getSpeedY ()

  Retorna a velocidade no eixo Y.

Public Member Functions inherited from GameObject

- Point getPos ()

  Retorna a posição do objeto.
- float getPosX ()

  Retorna a coordenada X do objeto.
- float getPosY ()

  Retorna a coordenada Y do objeto.
- virtual bool updatePosition ()=0

  Atualiza a posição do objeto.

Protected Member Functions

- Drawable (const Point &pos, const Point &spd)

  Construtor protegido com posição e velocidade.
- Drawable (const Point &pos)

  Construtor protegido apenas com posição.
- void setSpeedX (float x)

  Define a velocidade no eixo X.
- void setSpeedY (float y)

  Define a velocidade no eixo Y.
- void setSpeed (const Point &pos)

  Define o vetor de velocidade.

Protected Member Functions inherited from GameObject

- GameObject (const Point &position)

  Construtor protegido com posição.
- GameObject (float pX, float pY)

  Construtor protegido com coordenadas.
- void setPosX (float x)

  Define a coordenada X do objeto.
- void setPosY (float y)

  Define a coordenada Y do objeto.
- void setPos (const Point &p)

  Define a posição do objeto.

4.6.1 Detailed Description

Classe base para todos os objetos desenháveis e que podem se mover.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Drawable() [1/2]

```
Drawable::Drawable (
            const Point & pos,
            const Point & spd)  [protected]
```

Construtor protegido com posição e velocidade.

Parameters

pos	Posição inicial.
spd	Velocidade inicial.

4.6.2.2 Drawable() [2/2]

```
Drawable::Drawable (
            const Point & pos)  [protected]
```

Construtor protegido apenas com posição.

Parameters

pos	Posição inicial.

4.6.3 Member Function Documentation

4.6.3.1 addSpeedVector()

```
void Drawable::addSpeedVector (
            const Point & spd)
```

Adiciona um vetor à velocidade atual.

Parameters

spd	Vetor a ser adicionado.

4.6.3.2 addSpeedX()

```
void Drawable::addSpeedX (
            float x)
```

Adiciona um valor à velocidade X.

Parameters

x	Valor a ser adicionado.

4.6.3.3 addSpeedY()

```
void Drawable::addSpeedY (
            float y)
```

Adiciona um valor à velocidade Y.

Parameters

y	Valor a ser adicionado.

4.6.3.4 draw()

```
virtual void Drawable::draw ()  [pure virtual]
```

Desenha o objeto na tela.

Implemented in Background, Eel, Pipe, Player, and TransitionScreen.

4.6.3.5 getSpeed()

```
Point Drawable::getSpeed ()
```

Retorna o vetor de velocidade.

Returns

Vetor de velocidade.

4.6.3.6 getSpeedX()

```
float Drawable::getSpeedX ()
```

Retorna a velocidade no eixo X.

Returns

Velocidade X.

4.6.3.7 getSpeedY()

```
float Drawable::getSpeedY ()
```

Retorna a velocidade no eixo Y.

Returns

Velocidade Y.

4.6.3.8 setSpeed()

```
void Drawable::setSpeed (
            const Point & pos)  [protected]
```

Define o vetor de velocidade.

Parameters

pos	Novo vetor de velocidade.

4.6.3.9 setSpeedX()

```
void Drawable::setSpeedX (
            float x)  [protected]
```

Define a velocidade no eixo X.

Parameters

x	Nova velocidade X.

4.6.3.10 setSpeedY()

```
void Drawable::setSpeedY (
            float y)  [protected]
```

Define a velocidade no eixo Y.

Parameters

y	Nova velocidade Y.

4.6.3.11 updateSpeed()

```
virtual void Drawable::updateSpeed ()  [pure virtual]
```

Atualiza a velocidade do objeto.

Implemented in Background, Pipe, Player, and TransitionScreen.

The documentation for this class was generated from the following files:

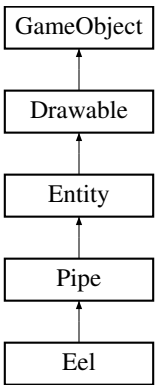
- include/game_object.hpp
- src/game_object.cpp

4.7 Eel Class Reference

Classe para obstáculos do tipo Eel (enguia).

```
#include <entity.hpp>
```

Inheritance diagram for Eel:



Public Member Functions

- Eel (const Point &pos, Spritesheet *image)
  Construtor da Eel.
- void **rotate** ()
  Rotaciona a enguia.
- bool updatePosition () override
  Atualiza a posição da enguia.
- void draw () override
  Desenha a enguia na tela.

Public Member Functions inherited from Pipe

- Pipe (const Point &pos, float w, float h, ALLEGRO_BITMAP *image=nullptr, bool inv=false)

  Construtor do Pipe.
- bool updatePosition () override

  Atualiza a posição do obstáculo.
- void updateSpeed () override

  Atualiza a velocidade do obstáculo.
- bool loadSprite ()

  Carrega o sprite do obstáculo.
- void draw () override

  Desenha o obstáculo na tela.

Public Member Functions inherited from Entity

- Entity (const Point &pos, const Point &spd)

  Construtor da entidade com posição e velocidade.
- Entity (const Point &pos)

  Construtor da entidade apenas com posição.
- Hitbox * getHitbox ()

  Retorna a hitbox da entidade.
- virtual ~**Entity** ()

  Destrutor virtual.

Public Member Functions inherited from Drawable

- void addSpeedVector (const Point &spd)

  Adiciona um vetor à velocidade atual.
- void addSpeedX (float x)

  Adiciona um valor à velocidade X.
- void addSpeedY (float y)

  Adiciona um valor à velocidade Y.
- Point getSpeed ()

  Retorna o vetor de velocidade.
- float getSpeedX ()

  Retorna a velocidade no eixo X.
- float getSpeedY ()

  Retorna a velocidade no eixo Y.

Public Member Functions inherited from GameObject

- Point getPos ()

  Retorna a posição do objeto.
- float getPosX ()

  Retorna a coordenada X do objeto.
- float getPosY ()

  Retorna a coordenada Y do objeto.

Additional Inherited Members

Static Public Member Functions inherited from Pipe

- static void updateScreenSpeed (float s)

  Atualiza a velocidade global dos obstáculos.

Static Public Attributes inherited from Pipe

- static float **screenSpeed** = -7

  Velocidade global dos obstáculos.

Protected Member Functions inherited from Drawable

- Drawable (const Point &pos, const Point &spd)

  Construtor protegido com posição e velocidade.
- Drawable (const Point &pos)

  Construtor protegido apenas com posição.
- void setSpeedX (float x)

  Define a velocidade no eixo X.
- void setSpeedY (float y)

  Define a velocidade no eixo Y.
- void setSpeed (const Point &pos)

  Define o vetor de velocidade.

Protected Member Functions inherited from GameObject

- GameObject (const Point &position)

  Construtor protegido com posição.
- GameObject (float pX, float pY)

  Construtor protegido com coordenadas.
- void setPosX (float x)

  Define a coordenada X do objeto.
- void setPosY (float y)

  Define a coordenada Y do objeto.
- void setPos (const Point &p)

  Define a posição do objeto.

Protected Attributes inherited from Entity

- Hitbox * **hb**

  Ponteiro para a hitbox da entidade.

4.7.1 Detailed Description

Classe para obstáculos do tipo Eel (enguia).

4.7.2 Constructor & Destructor Documentation

4.7.2.1 Eel()

```
Eel::Eel (
            const Point & pos,
            Spritesheet * image)
```

Construtor da Eel.

Parameters

pos	Posição inicial.
image	Spritesheet da enguia.

4.7.3 Member Function Documentation

4.7.3.1 draw()

```
void Eel::draw ()  [override], [virtual]
```

Desenha a enguia na tela.

Implements Drawable.

4.7.3.2 updatePosition()

```
bool Eel::updatePosition ()  [override], [virtual]
```

Atualiza a posição da enguia.

Returns

true se a posição foi atualizada.

Implements GameObject.

The documentation for this class was generated from the following files:

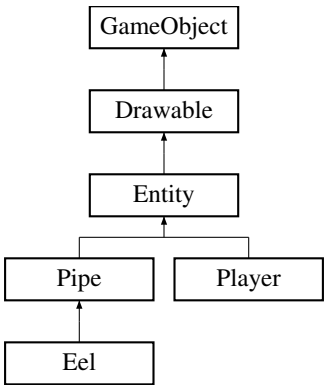
- include/entity.hpp
- src/entity.cpp

4.8 Entity Class Reference

Classe base para objetos do jogo que podem interagir (possuem hitbox).

```
#include <entity.hpp>
```

Inheritance diagram for Entity:



Public Member Functions

- Entity (const Point &pos, const Point &spd)

  Construtor da entidade com posição e velocidade.
- Entity (const Point &pos)

  Construtor da entidade apenas com posição.
- Hitbox * getHitbox ()

  Retorna a hitbox da entidade.
- virtual ~**Entity** ()

  Destrutor virtual.

Public Member Functions inherited from Drawable

- virtual void draw ()=0

  Desenha o objeto na tela.
- virtual void updateSpeed ()=0

  Atualiza a velocidade do objeto.
- void addSpeedVector (const Point &spd)

  Adiciona um vetor à velocidade atual.
- void addSpeedX (float x)

  Adiciona um valor à velocidade X.
- void addSpeedY (float y)

  Adiciona um valor à velocidade Y.
- Point getSpeed ()

  Retorna o vetor de velocidade.
- float getSpeedX ()

  Retorna a velocidade no eixo X.
- float getSpeedY ()

  Retorna a velocidade no eixo Y.

Public Member Functions inherited from GameObject

- Point getPos ()

  Retorna a posição do objeto.
- float getPosX ()

  Retorna a coordenada X do objeto.
- float getPosY ()

  Retorna a coordenada Y do objeto.
- virtual bool updatePosition ()=0

  Atualiza a posição do objeto.

Protected Attributes

- Hitbox * **hb**

  Ponteiro para a hitbox da entidade.

Additional Inherited Members

Protected Member Functions inherited from Drawable

- Drawable (const Point &pos, const Point &spd)

  Construtor protegido com posição e velocidade.
- Drawable (const Point &pos)

  Construtor protegido apenas com posição.
- void setSpeedX (float x)

  Define a velocidade no eixo X.
- void setSpeedY (float y)

  Define a velocidade no eixo Y.
- void setSpeed (const Point &pos)

  Define o vetor de velocidade.

Protected Member Functions inherited from GameObject

- GameObject (const Point &position)

  Construtor protegido com posição.
- GameObject (float pX, float pY)

  Construtor protegido com coordenadas.
- void setPosX (float x)

  Define a coordenada X do objeto.
- void setPosY (float y)

  Define a coordenada Y do objeto.
- void setPos (const Point &p)

  Define a posição do objeto.

4.8.1 Detailed Description

Classe base para objetos do jogo que podem interagir (possuem hitbox).

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Entity() [1/2]

```
Entity::Entity (
            const Point & pos,
            const Point & spd)
```

Construtor da entidade com posição e velocidade.

Parameters

pos	Posição inicial.
spd	Velocidade inicial.

4.8.2.2 Entity() [2/2]

```
Entity::Entity (
            const Point & pos)
```

Construtor da entidade apenas com posição.

Parameters

pos	Posição inicial.

4.8.3 Member Function Documentation

4.8.3.1 getHitbox()

```
Hitbox * Entity::getHitbox ()
```

Retorna a hitbox da entidade.

Returns

Ponteiro para a hitbox.

The documentation for this class was generated from the following files:

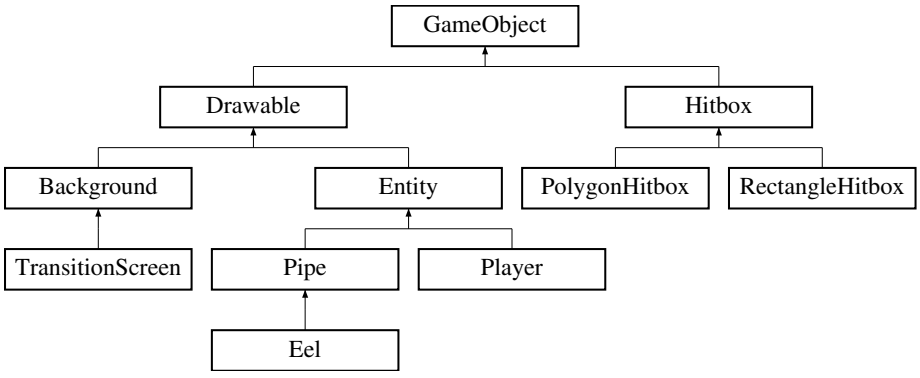
- include/entity.hpp
- src/entity.cpp

4.9 GameObject Class Reference

Classe base para todos os objetos do jogo.

```
#include <game_object.hpp>
```

Inheritance diagram for GameObject:



Public Member Functions

- Point getPos ()
  Retorna a posição do objeto.
- float getPosX ()
  Retorna a coordenada X do objeto.
- float getPosY ()
  Retorna a coordenada Y do objeto.
- virtual bool updatePosition ()=0
  Atualiza a posição do objeto.

Protected Member Functions

- GameObject (const Point &position)

  Construtor protegido com posição.
- GameObject (float pX, float pY)

  Construtor protegido com coordenadas.
- void setPosX (float x)

  Define a coordenada X do objeto.
- void setPosY (float y)

  Define a coordenada Y do objeto.
- void setPos (const Point &p)

  Define a posição do objeto.

4.9.1 Detailed Description

Classe base para todos os objetos do jogo.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 GameObject() [1/2]

```
GameObject::GameObject (
            const Point & position)  [protected]
```

Construtor protegido com posição.

Parameters

position	Posição inicial do objeto.

4.9.2.2 GameObject() [2/2]

```
GameObject::GameObject (
            float pX,
            float pY)  [protected]
```

Construtor protegido com coordenadas.

Parameters

pX	Coordenada X.
pY	Coordenada Y.

4.9.3 Member Function Documentation

4.9.3.1 getPos()

```
Point GameObject::getPos ()
```

Retorna a posição do objeto.

Returns

Posição atual.

4.9.3.2 getPosX()

```
float GameObject::getPosX ()
```

Retorna a coordenada X do objeto.

Returns

Coordenada X.

4.9.3.3 getPosY()

```
float GameObject::getPosY ()
```

Retorna a coordenada Y do objeto.

Returns

Coordenada Y.

4.9.3.4 setPos()

```
void GameObject::setPos (
            const Point & p)  [protected]
```

Define a posição do objeto.

Parameters

p	Nova posição.

4.9.3.5 setPosX()

```
void GameObject::setPosX (
            float x)  [protected]
```

Define a coordenada X do objeto.

Parameters

x	Nova coordenada X.

4.9.3.6 setPosY()

```
void GameObject::setPosY (
            float y)  [protected]
```

Define a coordenada Y do objeto.

Parameters

y	Nova coordenada Y.

4.9.3.7 updatePosition()

```
virtual bool GameObject::updatePosition ()  [pure virtual]
```

Atualiza a posição do objeto.

Returns

true se a posição foi atualizada.

Implemented in Background, Eel, Hitbox, Pipe, Player, PolygonHitbox, RectangleHitbox, and TransitionScreen.

The documentation for this class was generated from the following files:

- include/game_object.hpp
- src/game_object.cpp

4.10 Handler Class Reference

Public Member Functions

- int gameOn (ALLEGRO_TIMER &timer, ALLEGRO_TIMER &animation_timer, ALLEGRO_EVENT_QUEUE &eventQueue, const int SCREEN_H, const int SCREEN_W)

  Executa o loop principal do jogo.
- void addObstacle (ALLEGRO_BITMAP *image, Spritesheet *eelImage)

  Adiciona um novo obstáculo ao jogo.
- bool outOfBorders ()

  Verifica se o jogador saiu dos limites da tela.
- bool checkCollisions ()

  Verifica colisões entre o jogador e os obstáculos.
- void **drawAll** ()

  Desenha todos os objetos do jogo na tela.
- void **death** ()

  Executa procedimentos de morte do jogador.
- int sortBetween (int x, int y)

  Sorteia um número inteiro entre min e max.
- void **updateAmbient** ()

  Atualiza o ambiente do jogo, mudando dinâmicas e velocidade.
- void **drawObstacles** ()

4.10.1 Member Function Documentation

4.10.1.1 addObstacle()

```
void Handler::addObstacle (
            ALLEGRO_BITMAP * image,
            Spritesheet * eelImage)
```

Adiciona um novo obstáculo ao jogo.

Parameters

image	Bitmap do obstáculo tipo Pipe.
eelImage	Spritesheet do obstáculo tipo Eel.

4.10.1.2 checkCollisions()

```
bool Handler::checkCollisions ()
```

Verifica colisões entre o jogador e os obstáculos.

Returns

true Se houve colisão.

false Caso contrário.

4.10.1.3 gameOn()

```
int Handler::gameOn (
            ALLEGRO_TIMER & timer,
            ALLEGRO_TIMER & animation_timer,
            ALLEGRO_EVENT_QUEUE & eventQueue,
            const int SCREEN_H,
            const int SCREEN_W)
```

Executa o loop principal do jogo.

Parameters

timer	Timer principal do jogo.
animation_timer	Timer para animações.
eventQueue	Fila de eventos do Allegro.
SCREEN_H	Altura da tela.
SCREEN_W	Largura da tela.

Returns

int Pontuação final (tempo de sobrevivência).

4.10.1.4 outOfBorders()

```
bool Handler::outOfBorders ()
```

Verifica se o jogador saiu dos limites da tela.

Returns

true Se saiu dos limites.

false Caso contrário.

4.10.1.5 sortBetween()

```
int Handler::sortBetween (
            int x,
            int y)
```

Sorteia um número inteiro entre min e max.

Parameters

min	Valor mínimo.
max	Valor máximo.

Returns

int Número sorteado.

The documentation for this class was generated from the following files:

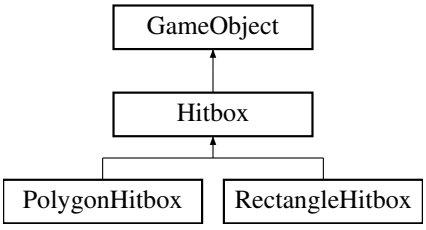
- include/game_object_handler.hpp
- src/game_object_handler.cpp

4.11 Hitbox Class Reference

Classe base para hitboxes de objetos do jogo.

```
#include <hitbox.hpp>
```

Inheritance diagram for Hitbox:



Public Member Functions

- Hitbox (const Point &center)
  Construtor da hitbox.
- bool updatePosition () override
  Atualiza a posição da hitbox.
- void setTarget (Drawable *target)
  Define o objeto alvo associado à hitbox.
- virtual Polygon getPolygon ()=0
  Retorna o polígono da hitbox.
- virtual float getAngle ()=0
  Retorna o ângulo da hitbox.
- virtual void rotateHitbox (float radians)=0
  Rotaciona a hitbox.
- virtual ~**Hitbox** ()
  Destrutor virtual da hitbox.

Public Member Functions inherited from GameObject

- Point getPos ()
  Retorna a posição do objeto.
- float getPosX ()
  Retorna a coordenada X do objeto.
- float getPosY ()
  Retorna a coordenada Y do objeto.

Additional Inherited Members

Protected Member Functions inherited from GameObject

- GameObject (const Point &position)
  Construtor protegido com posição.
- GameObject (float pX, float pY)
  Construtor protegido com coordenadas.
- void setPosX (float x)
  Define a coordenada X do objeto.
- void setPosY (float y)
  Define a coordenada Y do objeto.
- void setPos (const Point &p)
  Define a posição do objeto.

4.11.1 Detailed Description

Classe base para hitboxes de objetos do jogo.

Utilizada para determinar o espaço ocupado no jogo e interagir com outros objetos desenháveis.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 Hitbox()

```
Hitbox::Hitbox (
            const Point & center)
```

Construtor da hitbox.

Parameters

center	Centro da hitbox.

4.11.3 Member Function Documentation

4.11.3.1 getAngle()

```
virtual float Hitbox::getAngle ()  [pure virtual]
```

Retorna o ângulo da hitbox.

Returns

Ângulo em radianos.

Implemented in PolygonHitbox, and RectangleHitbox.

4.11.3.2 getPolygon()

```
virtual Polygon Hitbox::getPolygon ()  [pure virtual]
```

Retorna o polígono da hitbox.

Returns

Polígono da hitbox.

Implemented in PolygonHitbox, and RectangleHitbox.

4.11.3.3 rotateHitbox()

```
virtual void Hitbox::rotateHitbox (
            float radians)  [pure virtual]
```

Rotaciona a hitbox.

Parameters

radians	Ângulo em radianos.

Implemented in PolygonHitbox, and RectangleHitbox.

4.11.3.4 setTarget()

```
void Hitbox::setTarget (
            Drawable * target)
```

Define o objeto alvo associado à hitbox.

Parameters

target	Ponteiro para o objeto Drawable.

4.11.3.5 updatePosition()

```
bool Hitbox::updatePosition ()  [override], [virtual]
```

Atualiza a posição da hitbox.

Returns

true se a posição foi atualizada.

Implements GameObject.

Reimplemented in PolygonHitbox, and RectangleHitbox.

The documentation for this class was generated from the following files:

- include/hitbox.hpp
- src/hitbox.cpp

4.12 LeaderBoard Class Reference

Classe responsável por gerenciar e exibir o placar de líderes (leaderboard).

```
#include <leaderboard.hpp>
```

Public Member Functions

- LeaderBoard (string filename, RectangleT rect)
  Construtor do LeaderBoard.
- bool addNewProfile (Profile profile)
  Adiciona um novo perfil ao leaderboard.
- void **updateLeaderBoard** ()
  Atualiza o leaderboard com os melhores perfis.
- void setFirstRowColor (Color color)
  Define a cor da primeira linha.
- void setSecondRowColor (Color color)
  Define a cor da segunda linha.
- void setThirdRowColor (Color color)
  Define a cor da terceira linha.
- void setOthersRowsColor (Color color)
  Define a cor das demais linhas.
- void setFirstRowTextColor (Color color)
  Define a cor do texto da primeira linha.
- void setSecondRowTextColor (Color color)

Define a cor do texto da segunda linha.

- void setThirdRowTextColor (Color color)

  Define a cor do texto da terceira linha.

- void setOthersRowsTextColor (Color color)

  Define a cor do texto das demais linhas.

- void drawLeaderBoard (ALLEGRO_FONT *font)

  Desenha o leaderboard na tela.

- void **display** ()

  Exibe o leaderboard no console.

- void save (string filename)

  Salva o leaderboard em um arquivo.

Public Attributes

- Table **table**

  Tabela para exibição do leaderboard.

4.12.1 Detailed Description

Classe responsável por gerenciar e exibir o placar de líderes (leaderboard).

4.12.2 Constructor & Destructor Documentation

4.12.2.1 LeaderBoard()

```
LeaderBoard::LeaderBoard (
            string filename,
            RectangleT rect)
```

Construtor do LeaderBoard.

Parameters

filename	Nome do arquivo da base de dados.
rect	Retângulo para exibição da tabela.

4.12.3 Member Function Documentation

4.12.3.1 addNewProfile()

```
bool LeaderBoard::addNewProfile (
            Profile profile)
```

Adiciona um novo perfil ao leaderboard.

Parameters

profile	Perfil a ser adicionado.

Returns

true se adicionado com sucesso.

4.12.3.2 drawLeaderBoard()

```
void LeaderBoard::drawLeaderBoard (
            ALLEGRO_FONT * font)
```

Desenha o leaderboard na tela.

Parameters

font	Fonte a ser utilizada.

4.12.3.3 save()

```
void LeaderBoard::save (
            string filename)
```

Salva o leaderboard em um arquivo.

Parameters

filename	Nome do arquivo.

4.12.3.4 setFirstRowColor()

```
void LeaderBoard::setFirstRowColor (
            Color color)
```

Define a cor da primeira linha.

Parameters

color	Cor a ser definida.

4.12.3.5 setFirstRowTextColor()

```
void LeaderBoard::setFirstRowTextColor (
            Color color)
```

Define a cor do texto da primeira linha.

Parameters

color	Cor a ser definida.

4.12.3.6 setOthersRowsColor()

```
void LeaderBoard::setOthersRowsColor (
            Color color)
```

Define a cor das demais linhas.

Parameters

color	Cor a ser definida.

4.12.3.7 setOthersRowsTextColor()

```
void LeaderBoard::setOthersRowsTextColor (
            Color color)
```

Define a cor do texto das demais linhas.

Parameters

color	Cor a ser definida.

4.12.3.8 setSecondRowColor()

```
void LeaderBoard::setSecondRowColor (
            Color color)
```

Define a cor da segunda linha.

Parameters

color	Cor a ser definida.

4.12.3.9 setSecondRowTextColor()

```
void LeaderBoard::setSecondRowTextColor (
            Color color)
```

Define a cor do texto da segunda linha.

Parameters

color	Cor a ser definida.

4.12.3.10 setThirdRowColor()

```
void LeaderBoard::setThirdRowColor (
            Color color)
```

Define a cor da terceira linha.

Parameters

color	Cor a ser definida.

4.12.3.11 setThirdRowTextColor()

```
void LeaderBoard::setThirdRowTextColor (
            Color color)
```

Define a cor do texto da terceira linha.

Parameters

color	Cor a ser definida.

The documentation for this class was generated from the following files:

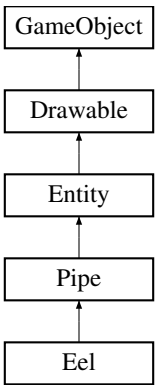
- include/leaderboard.hpp
- src/leaderboard.cpp

4.13 Pipe Class Reference

Classe básica para obstáculos do tipo Pipe.

```
#include <entity.hpp>
```

Inheritance diagram for Pipe:



Public Member Functions

- Pipe (const Point &pos, float w, float h, ALLEGRO_BITMAP *image=nullptr, bool inv=false)

  Construtor do Pipe.
- bool updatePosition () override

  Atualiza a posição do obstáculo.
- void updateSpeed () override

  Atualiza a velocidade do obstáculo.
- bool loadSprite ()

  Carrega o sprite do obstáculo.
- void draw () override

  Desenha o obstáculo na tela.

Public Member Functions inherited from Entity

- Entity (const Point &pos, const Point &spd)

  Construtor da entidade com posição e velocidade.
- Entity (const Point &pos)

  Construtor da entidade apenas com posição.
- Hitbox * getHitbox ()

  Retorna a hitbox da entidade.
- virtual ~**Entity** ()

  Destrutor virtual.

Public Member Functions inherited from Drawable

- void addSpeedVector (const Point &spd)

  Adiciona um vetor à velocidade atual.
- void addSpeedX (float x)

  Adiciona um valor à velocidade X.
- void addSpeedY (float y)

  Adiciona um valor à velocidade Y.
- Point getSpeed ()

  Retorna o vetor de velocidade.
- float getSpeedX ()

  Retorna a velocidade no eixo X.
- float getSpeedY ()

  Retorna a velocidade no eixo Y.

Public Member Functions inherited from GameObject

- Point getPos ()

  Retorna a posição do objeto.
- float getPosX ()

  Retorna a coordenada X do objeto.
- float getPosY ()

  Retorna a coordenada Y do objeto.

Static Public Member Functions

- static void updateScreenSpeed (float s)

  Atualiza a velocidade global dos obstáculos.

Static Public Attributes

- static float **screenSpeed** = -7

  Velocidade global dos obstáculos.

Additional Inherited Members

Protected Member Functions inherited from Drawable

- Drawable (const Point &pos, const Point &spd)

  Construtor protegido com posição e velocidade.
- Drawable (const Point &pos)

  Construtor protegido apenas com posição.
- void setSpeedX (float x)

  Define a velocidade no eixo X.
- void setSpeedY (float y)

  Define a velocidade no eixo Y.
- void setSpeed (const Point &pos)

  Define o vetor de velocidade.

Protected Member Functions inherited from GameObject

- GameObject (const Point &position)

  Construtor protegido com posição.
- GameObject (float pX, float pY)

  Construtor protegido com coordenadas.
- void setPosX (float x)

  Define a coordenada X do objeto.
- void setPosY (float y)

  Define a coordenada Y do objeto.
- void setPos (const Point &p)

  Define a posição do objeto.

Protected Attributes inherited from Entity

- Hitbox * **hb**

  Ponteiro para a hitbox da entidade.

4.13.1 Detailed Description

Classe básica para obstáculos do tipo Pipe.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 Pipe()

```
Pipe::Pipe (
            const Point & pos,
            float w,
            float h,
            ALLEGRO_BITMAP * image = nullptr,
            bool inv = false)
```

Construtor do Pipe.

Parameters

pos	Posição inicial.
w	Largura.
h	Altura.
image	Bitmap do obstáculo.
inv	Indica se está invertido.

4.13.3 Member Function Documentation

4.13.3.1 draw()

```
void Pipe::draw ()  [override], [virtual]
```

Desenha o obstáculo na tela.

Implements Drawable.

4.13.3.2 loadSprite()

```
bool Pipe::loadSprite ()
```

Carrega o sprite do obstáculo.

Returns

true se carregado com sucesso.

4.13.3.3 updatePosition()

```
bool Pipe::updatePosition ()  [override], [virtual]
```

Atualiza a posição do obstáculo.

Returns

true se a posição foi atualizada.

Implements GameObject.

4.13.3.4 updateScreenSpeed()

```
void Pipe::updateScreenSpeed (
            float s)  [static]
```

Atualiza a velocidade global dos obstáculos.

Parameters

s	Nova velocidade.

4.13.3.5 updateSpeed()

```
void Pipe::updateSpeed ()  [override], [virtual]
```

Atualiza a velocidade do obstáculo.

Implements Drawable.

The documentation for this class was generated from the following files:

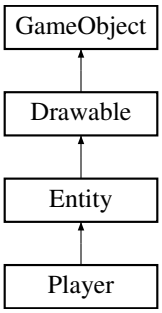
- include/entity.hpp
- src/entity.cpp

4.14 Player Class Reference

Classe que representa o jogador principal.

```
#include <entity.hpp>
```

Inheritance diagram for Player:



Public Member Functions

- **Player** ()
  Construtor padrão do jogador.
- bool updatePosition () override
  Atualiza a posição do jogador.
- void updateSpeed () override
  Atualiza a velocidade do jogador.
- void **jump** ()
  Realiza o pulo do jogador.
- void draw () override
  Desenha o jogador na tela.
- void **updatePlayerState** ()
  Atualiza o estado do jogador.
- void **updateAnimation** ()
  Atualiza a animação do jogador.
- void setPlayerState (PlayerState s)
  Define o estado do jogador.
- ~**Player** ()
  Destrutor do jogador.

Public Member Functions inherited from Entity

- Entity (const Point &pos, const Point &spd)

  Construtor da entidade com posição e velocidade.
- Entity (const Point &pos)

  Construtor da entidade apenas com posição.
- Hitbox * getHitbox ()

  Retorna a hitbox da entidade.
- virtual ~**Entity** ()

  Destrutor virtual.

Public Member Functions inherited from Drawable

- void addSpeedVector (const Point &spd)

  Adiciona um vetor à velocidade atual.
- void addSpeedX (float x)

  Adiciona um valor à velocidade X.
- void addSpeedY (float y)

  Adiciona um valor à velocidade Y.
- Point getSpeed ()

  Retorna o vetor de velocidade.
- float getSpeedX ()

  Retorna a velocidade no eixo X.
- float getSpeedY ()

  Retorna a velocidade no eixo Y.

Public Member Functions inherited from GameObject

- Point getPos ()

  Retorna a posição do objeto.
- float getPosX ()

  Retorna a coordenada X do objeto.
- float getPosY ()

  Retorna a coordenada Y do objeto.

Additional Inherited Members

Protected Member Functions inherited from Drawable

- Drawable (const Point &pos, const Point &spd)

  Construtor protegido com posição e velocidade.
- Drawable (const Point &pos)

  Construtor protegido apenas com posição.
- void setSpeedX (float x)

  Define a velocidade no eixo X.
- void setSpeedY (float y)

  Define a velocidade no eixo Y.
- void setSpeed (const Point &pos)

  Define o vetor de velocidade.

Protected Member Functions inherited from GameObject

- GameObject (const Point &position)

  Construtor protegido com posição.
- GameObject (float pX, float pY)

  Construtor protegido com coordenadas.
- void setPosX (float x)

  Define a coordenada X do objeto.
- void setPosY (float y)

  Define a coordenada Y do objeto.
- void setPos (const Point &p)

  Define a posição do objeto.

Protected Attributes inherited from Entity

- Hitbox * **hb**

  Ponteiro para a hitbox da entidade.

4.14.1 Detailed Description

Classe que representa o jogador principal.

4.14.2 Member Function Documentation

4.14.2.1 draw()

```
void Player::draw ()  [override], [virtual]
```

Desenha o jogador na tela.

Implements Drawable.

4.14.2.2 setPlayerState()

```
void Player::setPlayerState (
            PlayerState s)
```

Define o estado do jogador.

Parameters

s	Novo estado.

4.14.2.3 updatePosition()

```
bool Player::updatePosition ()  [override], [virtual]
```

Atualiza a posição do jogador.

Returns

true se a posição foi atualizada.

Implements GameObject.

4.14.2.4 updateSpeed()

```
void Player::updateSpeed ()  [override], [virtual]
```

Atualiza a velocidade do jogador.

Implements Drawable.

The documentation for this class was generated from the following files:

- include/entity.hpp
- src/entity.cpp

4.15 Point Struct Reference

Estrutura que representa um ponto ou vetor 2D.

`#include <polygon.hpp>`

Public Member Functions

- Point (float x, float y)
  Construtor que inicializa as coordenadas.
- **Point** ()
  Construtor padrão (x=0, y=0).
- Point **operator+** (const Point &p2) const
  Soma de pontos/vetores.
- Point **operator-** (const Point &p2) const
  Subtração de pontos/vetores.
- bool **operator<** (const Point &p2) const
  Operador de comparação para ordenação.
- Point **getNormalVector** () const
  Retorna o vetor normal ao vetor atual.
- float **dotProduct** (const Point &p2) const
  Produto escalar entre dois vetores.
- Point **normalizeVector** () const
  Normaliza o vetor.
- Point rotateVector (float angle) const
  Rotaciona o vetor por um ângulo.
- Point **rotateVector** (float cosA, float sinA) const
  Rotaciona o vetor usando cosseno e seno.
- Point rotatePoint (const Point &rotationCenter, float angle) const
  Rotaciona o ponto em torno de um centro.
- Point **rotatePoint** (const Point &rotationCenter, float cosA, float sinA) const
  Rotaciona o ponto em torno de um centro usando cosseno e seno.

Static Public Member Functions

- static Point **getEdgeVector** (const Point &p1, const Point &p2)

  Retorna o vetor aresta entre dois pontos.

Public Attributes

- float **x**
- float **y**

4.15.1 Detailed Description

Estrutura que representa um ponto ou vetor 2D.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 Point()

```
Point::Point (
            float x,
            float y)  [inline]
```

Construtor que inicializa as coordenadas.

Parameters

x	Coordenada X.
y	Coordenada Y.

4.15.3 Member Function Documentation

4.15.3.1 rotatePoint()

```
Point Point::rotatePoint (
            const Point & rotationCenter,
            float angle) const
```

Rotaciona o ponto em torno de um centro.

Parameters

rotationCenter	Centro de rotação.
angle	Ângulo em radianos.

4.15.3.2 rotateVector()

```
Point Point::rotateVector (
            float angle) const
```

Rotaciona o vetor por um ângulo.

Parameters

angle	Ângulo em radianos.

Returns

Vetor rotacionado.

The documentation for this struct was generated from the following files:

- include/polygon.hpp
- src/polygon.cpp

4.16 PointT Struct Reference

Estrutura que representa um ponto 2D para tabelas.

`#include <table.hpp>`

Public Member Functions

- **PointT** ()
  Construtor padrão.
- PointT (float x, float y)
  Construtor com coordenadas.
- PointT (const PointT &other)
  Construtor de cópia.
- PointT & operator= (const PointT &other)
  Operador de atribuição.
- PointT operator- (const PointT &other) const
  Subtrai dois pontos.
- PointT operator+ (const PointT &other) const
  Soma dois pontos.
- PointT operator/ (float f) const
  Divide ponto por escalar.
- PointT operator∗ (float f)
  Multiplica ponto por escalar.
- void **display** ()
  Exibe o ponto no console.

Public Attributes

- float **x**
- float **y**
  Coordenadas do ponto.

4.16.1 Detailed Description

Estrutura que representa um ponto 2D para tabelas.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 PointT() [1/2]

```
PointT::PointT (
            float x,
            float y)
```

Construtor com coordenadas.

Parameters

x	Coordenada X.
y	Coordenada Y.

4.16.2.2 PointT() [2/2]

```
PointT::PointT (
            const PointT & other)
```

Construtor de cópia.

Parameters

other	Outro ponto.

4.16.3 Member Function Documentation

4.16.3.1 operator*()

```
PointT PointT::operator* (
            float f)
```

Multiplica ponto por escalar.

Parameters

f	Escalar.

Returns

Resultado da multiplicação.

4.16.3.2 operator+()

```
PointT PointT::operator+ (
            const PointT & other) const
```

Soma dois pontos.

Parameters

other	Outro ponto.

Returns

Resultado da soma.

4.16.3.3 operator-()

```
PointT PointT::operator- (
            const PointT & other) const
```

Subtrai dois pontos.

Parameters

other	Outro ponto.

Returns

Resultado da subtração.

4.16.3.4 operator/()

```
PointT PointT::operator/ (
            float f) const
```

Divide ponto por escalar.

Parameters

f	Escalar.

Returns

Resultado da divisão.

4.16.3.5 operator=()

```
PointT & PointT::operator= (
            const PointT & other)
```

Operador de atribuição.

Parameters

other	Outro ponto.

Returns

Referência para este ponto.

The documentation for this struct was generated from the following files:

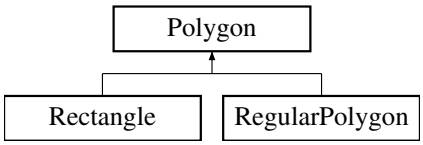
- include/table.hpp
- src/table.cpp

4.17 Polygon Struct Reference

Estrutura que representa um polígono genérico.

`#include <polygon.hpp>`

Inheritance diagram for Polygon:



Public Member Functions

- vector< Point > getRotatedVertices (const Point &center)
  Retorna os vértices rotacionados em torno de um centro.
- set< Point > **getEdgeVectors** ()
  Retorna o conjunto de vetores das arestas.
- set< Point > **getEdgeNormals** ()
  Retorna o conjunto de vetores normais das arestas.
- virtual Polygon getPolygon (const Point &center=Point(0, 0))
  Retorna o polígono (possivelmente modificado).
- **Polygon** (initializer_list< Point > vert)
  Construtor a partir de lista de vértices.
- **Polygon** (const std::vector< Point > &vert)
  Construtor a partir de vetor de vértices.
- **Polygon** (vector< Point > vert, int n, set< Point > vect, set< Point > normals)
  Construtor completo.
- **Polygon** (const Polygon &copy)
  Construtor de cópia.
- void **operator=** (const Polygon &copy)
  Operador de atribuição.
- float * **getPointArray** ()
  Retorna um array de floats com os pontos do polígono.
- void addAngle (float radians)
  Adiciona um ângulo de rotação ao polígono.
- virtual void updateVertices (const Point &delta)
  Atualiza os vértices do polígono.

Public Attributes

- vector< Point > **vertices**
- int **vertexCount**
- float **angle** = 0
- set< Point > **edgeVectors**
- set< Point > **edgeNormals**

4.17.1 Detailed Description

Estrutura que representa um polígono genérico.

4.17.2 Member Function Documentation

4.17.2.1 addAngle()

```
void Polygon::addAngle (
            float radians)
```

Adiciona um ângulo de rotação ao polígono.

Parameters

radians	Ângulo em radianos.

4.17.2.2 getPolygon()

```
Polygon Polygon::getPolygon (
            const Point & center = Point(0, 0))  [virtual]
```

Retorna o polígono (possivelmente modificado).

Parameters

center	Centro de referência.

Reimplemented in Rectangle, and RegularPolygon.

4.17.2.3 getRotatedVertices()

```
vector< Point > Polygon::getRotatedVertices (
            const Point & center)
```

Retorna os vértices rotacionados em torno de um centro.

Parameters

center	Centro de rotação.

4.17.2.4 updateVertices()

```
void Polygon::updateVertices (
            const Point & delta)  [virtual]
```

Atualiza os vértices do polígono.

Parameters

delta	Deslocamento.

Reimplemented in Rectangle, and RegularPolygon.

The documentation for this struct was generated from the following files:

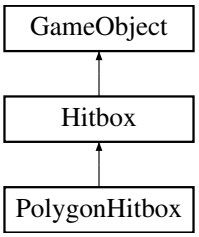
- include/polygon.hpp
- src/polygon.cpp

4.18 PolygonHitbox Class Reference

Hitbox com formato de polígono regular.

`#include <hitbox.hpp>`

Inheritance diagram for PolygonHitbox:



Public Member Functions

- PolygonHitbox (const Point &center, int n, float EdgeLength)
  Construtor da hitbox poligonal.
- int getSideCount ()
  Retorna o número de lados do polígono.
- float getEdgeLength ()
  Retorna o tamanho da aresta.
- Polygon getPolygon () override
  Retorna o polígono da hitbox.
- float getAngle () override
  Retorna o ângulo da hitbox.
- bool updatePosition () override
  Atualiza a posição da hitbox.
- void rotateHitbox (float radians) override
  Rotaciona a hitbox.
- float * getVertices ()
  Retorna os vértices do polígono.

Public Member Functions inherited from Hitbox

- Hitbox (const Point &center)

  Construtor da hitbox.
- void setTarget (Drawable *target)

  Define o objeto alvo associado à hitbox.
- virtual ~**Hitbox** ()

  Destrutor virtual da hitbox.

Public Member Functions inherited from GameObject

- Point getPos ()

  Retorna a posição do objeto.
- float getPosX ()

  Retorna a coordenada X do objeto.
- float getPosY ()

  Retorna a coordenada Y do objeto.

Additional Inherited Members

Protected Member Functions inherited from GameObject

- GameObject (const Point &position)

  Construtor protegido com posição.
- GameObject (float pX, float pY)

  Construtor protegido com coordenadas.
- void setPosX (float x)

  Define a coordenada X do objeto.
- void setPosY (float y)

  Define a coordenada Y do objeto.
- void setPos (const Point &p)

  Define a posição do objeto.

4.18.1 Detailed Description

Hitbox com formato de polígono regular.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 PolygonHitbox()

```
PolygonHitbox::PolygonHitbox (
            const Point & center,
            int n,
            float EdgeLength)
```

Construtor da hitbox poligonal.

Parameters

center	Centro da hitbox.
n	Número de lados.
EdgeLength	Tamanho da aresta.

4.18.3 Member Function Documentation

4.18.3.1 getAngle()

```
float PolygonHitbox::getAngle ()  [override], [virtual]
```

Retorna o ângulo da hitbox.

Returns

Ângulo em radianos.

Implements Hitbox.

4.18.3.2 getEdgeLength()

```
float PolygonHitbox::getEdgeLength ()
```

Retorna o tamanho da aresta.

Returns

Tamanho da aresta.

4.18.3.3 getPolygon()

```
Polygon PolygonHitbox::getPolygon ()  [override], [virtual]
```

Retorna o polígono da hitbox.

Returns

Polígono.

Implements Hitbox.

4.18.3.4 getSideCount()

```
int PolygonHitbox::getSideCount ()
```

Retorna o número de lados do polígono.

Returns

Número de lados.

4.18.3.5 getVertices()

```
float * PolygonHitbox::getVertices ()
```

Retorna os vértices do polígono.

Returns

Ponteiro para array de vértices.

4.18.3.6 rotateHitbox()

```
void PolygonHitbox::rotateHitbox (
            float radians)  [override], [virtual]
```

Rotaciona a hitbox.

Parameters

radians	Ângulo em radianos.

Implements Hitbox.

4.18.3.7 updatePosition()

```
bool PolygonHitbox::updatePosition ()  [override], [virtual]
```

Atualiza a posição da hitbox.

Returns

true se a posição foi atualizada.

Reimplemented from Hitbox.

The documentation for this class was generated from the following files:

- include/hitbox.hpp
- src/hitbox.cpp

4.19 PolygonProjection Struct Reference

Estrutura para projeção de polígono em um eixo (usada no SAT).

```
#include <polygon.hpp>
```

Public Member Functions

- PolygonProjection (const Polygon &poly, const Point &projAxis)
  Construtor que projeta um polígono em um eixo.
- bool doProjectionOverlap (const PolygonProjection &other)
  Verifica se há sobreposição entre duas projeções.

Public Attributes

- float **minProj**
- float **maxProj**

4.19.1 Detailed Description

Estrutura para projeção de polígono em um eixo (usada no SAT).

4.19.2 Constructor & Destructor Documentation

4.19.2.1 PolygonProjection()

```
PolygonProjection::PolygonProjection (
            const Polygon & poly,
            const Point & projAxis)  [inline]
```

Construtor que projeta um polígono em um eixo.

Parameters

poly	Polígono.
projAxis	Eixo de projeção.

4.19.3 Member Function Documentation

4.19.3.1 doProjectionOverlap()

```
bool PolygonProjection::doProjectionOverlap (
            const PolygonProjection & other)  [inline]
```

Verifica se há sobreposição entre duas projeções.

Parameters

other	Outra projeção.

Returns

true se houver sobreposição.

The documentation for this struct was generated from the following file:

- include/polygon.hpp

4.20 Profile Class Reference

Classe que representa o perfil de um jogador.

`#include <base.hpp>`

Public Member Functions

- **Profile** ()
  Construtor padrão.
- Profile (string, string)
  Construtor para novo jogador registrado.
- Profile (string, string, int, int)
  Construtor para inicializar lista de jogadores.
- Profile (const Profile &)
  Construtor de cópia.
- bool operator> (const Profile &) const
  Operador de comparação maior.
- bool operator< (const Profile &) const
  Operador de comparação menor.
- bool operator== (const Profile &) const
  Operador de igualdade.
- Profile operator= (const Profile &)
  Operador de atribuição.
- string **getName** ()
  Retorna o nome do jogador.
- string **getNickname** ()
  Retorna o nickname do jogador.
- int **getPlays** ()
  Retorna o número de partidas jogadas.
- int **getMaxDistance** ()
  Retorna a maior distância alcançada.
- void **setPlays** (int)
  Define o número de partidas jogadas.
- void **setMaxDistance** (int)
  Define a maior distância alcançada.
- void **display** ()
  Exibe as informações do perfil.

4.20.1 Detailed Description

Classe que representa o perfil de um jogador.

4.20.2 Constructor & Destructor Documentation

4.20.2.1 Profile() [1/3]

```
Profile::Profile (
            string name,
            string nickname)
```

Construtor para novo jogador registrado.

Parameters

name	Nome do jogador.
nickname	Apelido do jogador.

4.20.2.2 Profile() [2/3]

```
Profile::Profile (
            string name,
            string nickname,
            int maxDistance,
            int plays)
```

Construtor para inicializar lista de jogadores.

Parameters

name	Nome do jogador.
nickname	Apelido do jogador.
maxDistance	Maior distância.
plays	Número de partidas.

4.20.2.3 Profile() [3/3]

```
Profile::Profile (
            const Profile & otherProfile)
```

Construtor de cópia.

Parameters

other	Outro objeto Profile.

4.20.3 Member Function Documentation

4.20.3.1 operator<()

```
bool Profile::operator< (
            const Profile & otherProfile) const
```

Operador de comparação menor.

Parameters

other	Outro objeto Profile.

Returns

true se este perfil for menor.

4.20.3.2 operator=()

```
Profile Profile::operator= (
            const Profile & otherProfile)
```

Operador de atribuição.

Parameters

other	Outro objeto Profile.

Returns

Profile& Referência para este objeto.

4.20.3.3 operator==()

```
bool Profile::operator== (
            const Profile & otherProfile) const
```

Operador de igualdade.

Parameters

other	Outro objeto Profile.

Returns

true se os perfis forem iguais.

4.20.3.4 operator>()

```
bool Profile::operator> (
            const Profile & otherProfile) const
```

Operador de comparação maior.

Parameters

other	Outro objeto Profile.

Returns

true se este perfil for maior.

The documentation for this class was generated from the following files:

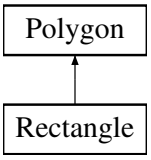
- include/base.hpp
- src/base.cpp

4.21 Rectangle Struct Reference

Estrutura que representa um retângulo (herda de Polygon).

`#include <polygon.hpp>`

Inheritance diagram for Rectangle:



Public Member Functions

- Rectangle (const Point &center, float w, float h)

  Construtor do retângulo.
- vector< Point > **getVertices** ()

  Retorna os vértices do retângulo.
- Polygon getPolygon (const Point &center=Point(0, 0)) override

  Retorna o polígono do retângulo.
- void updateVertices (const Point &delta) override

  Atualiza os vértices do retângulo.

Public Member Functions inherited from Polygon

- vector< Point > getRotatedVertices (const Point &center)

  Retorna os vértices rotacionados em torno de um centro.
- set< Point > **getEdgeVectors** ()

  Retorna o conjunto de vetores das arestas.
- set< Point > **getEdgeNormals** ()

  Retorna o conjunto de vetores normais das arestas.
- **Polygon** (initializer_list< Point > vert)

  Construtor a partir de lista de vértices.
- **Polygon** (const std::vector< Point > &vert)

  Construtor a partir de vetor de vértices.
- **Polygon** (vector< Point > vert, int n, set< Point > vect, set< Point > normals)

  Construtor completo.
- **Polygon** (const Polygon &copy)

  Construtor de cópia.
- void **operator=** (const Polygon &copy)

  Operador de atribuição.
- float * **getPointArray** ()

  Retorna um array de floats com os pontos do polígono.
- void addAngle (float radians)

  Adiciona um ângulo de rotação ao polígono.

Public Attributes

- float **width**
- float **height**
- Point **center**

Public Attributes inherited from Polygon

- vector< Point > **vertices**
- int **vertexCount**
- float **angle** = 0
- set< Point > **edgeVectors**
- set< Point > **edgeNormals**

4.21.1 Detailed Description

Estrutura que representa um retângulo (herda de Polygon).

4.21.2 Constructor & Destructor Documentation

4.21.2.1 Rectangle()

```
Rectangle::Rectangle (
            const Point & center,
            float w,
            float h)
```

Construtor do retângulo.

Parameters

center	Centro do retângulo.
w	Largura.
h	Altura.

4.21.3 Member Function Documentation

4.21.3.1 getPolygon()

```
Polygon Rectangle::getPolygon (
            const Point & center = Point(0, 0))  [override], [virtual]
```

Retorna o polígono do retângulo.

Reimplemented from Polygon.

4.21.3.2 updateVertices()

```
void Rectangle::updateVertices (
            const Point & delta)  [override], [virtual]
```

Atualiza os vértices do retângulo.

Reimplemented from Polygon.

The documentation for this struct was generated from the following files:

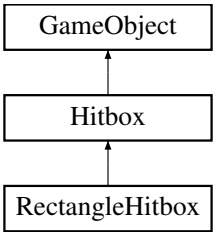
- include/polygon.hpp
- src/polygon.cpp

4.22 RectangleHitbox Class Reference

Hitbox retangular, usada para representar retângulos com largura e altura variáveis.

```
#include <hitbox.hpp>
```

Inheritance diagram for RectangleHitbox:



Public Member Functions

- RectangleHitbox (const Point &center, float w, float h)

  Construtor da hitbox retangular.
- float getWidth ()

  Retorna a largura da hitbox.
- float getHeight ()

  Retorna a altura da hitbox.
- Polygon getPolygon () override

  Retorna o polígono da hitbox.
- float getAngle () override

  Retorna o ângulo da hitbox.
- bool updatePosition () override

  Atualiza a posição da hitbox.
- void rotateHitbox (float radians) override

  Rotaciona a hitbox.
- float * getVertices ()

  Retorna os vértices do retângulo.

Public Member Functions inherited from Hitbox

- Hitbox (const Point &center)

  Construtor da hitbox.
- void setTarget (Drawable *target)

  Define o objeto alvo associado à hitbox.
- virtual ~**Hitbox** ()

  Destrutor virtual da hitbox.

Public Member Functions inherited from GameObject

- Point getPos ()

  Retorna a posição do objeto.
- float getPosX ()

  Retorna a coordenada X do objeto.
- float getPosY ()

  Retorna a coordenada Y do objeto.

Additional Inherited Members

Protected Member Functions inherited from GameObject

- GameObject (const Point &position)

  Construtor protegido com posição.
- GameObject (float pX, float pY)

  Construtor protegido com coordenadas.
- void setPosX (float x)

  Define a coordenada X do objeto.
- void setPosY (float y)

  Define a coordenada Y do objeto.
- void setPos (const Point &p)

  Define a posição do objeto.

4.22.1 Detailed Description

Hitbox retangular, usada para representar retângulos com largura e altura variáveis.

4.22.2 Constructor & Destructor Documentation

4.22.2.1 RectangleHitbox()

```
RectangleHitbox::RectangleHitbox (
            const Point & center,
            float w,
            float h)
```

Construtor da hitbox retangular.

Parameters

center	Centro da hitbox.
w	Largura.
h	Altura.

4.22.3 Member Function Documentation

4.22.3.1 getAngle()

```
float RectangleHitbox::getAngle ()  [override], [virtual]
```

Retorna o ângulo da hitbox.

Returns

Ângulo em radianos.

Implements Hitbox.

4.22.3.2 getHeight()

```
float RectangleHitbox::getHeight ()
```

Retorna a altura da hitbox.

Returns

Altura.

4.22.3.3 getPolygon()

```
Polygon RectangleHitbox::getPolygon ()  [override], [virtual]
```

Retorna o polígono da hitbox.

Returns

Polígono.

Implements Hitbox.

4.22.3.4 getVertices()

```
float * RectangleHitbox::getVertices ()
```

Retorna os vértices do retângulo.

Returns

Ponteiro para array de vértices.

4.22.3.5 getWidth()

```
float RectangleHitbox::getWidth ()
```

Retorna a largura da hitbox.

Returns

Largura.

4.22.3.6 rotateHitbox()

```
void RectangleHitbox::rotateHitbox (
            float radians)  [override], [virtual]
```

Rotaciona a hitbox.

Parameters

radians	Ângulo em radianos.

Implements Hitbox.

4.22.3.7 updatePosition()

```
bool RectangleHitbox::updatePosition ()  [override], [virtual]
```

Atualiza a posição da hitbox.

Returns

true se a posição foi atualizada.

Reimplemented from Hitbox.

The documentation for this class was generated from the following files:

- include/hitbox.hpp
- src/hitbox.cpp

4.23 RectangleT Struct Reference

Estrutura que representa um retângulo para tabelas.

```
#include <table.hpp>
```

Public Member Functions

- **RectangleT** ()
  Construtor padrão.
- RectangleT (PointT center, float length, float height)
  Construtor com centro, comprimento e altura.
- RectangleT (PointT topLeft, PointT bottomRight)
  Construtor com dois pontos.
- RectangleT (const RectangleT &other)
  Construtor de cópia.
- RectangleT & operator= (const RectangleT &other)
  Operador de atribuição.
- void **display** ()
  Exibe o retângulo no console.

Public Attributes

- PointT **center**
- PointT **topLeft**
- PointT **bottomRight**
  Pontos principais do retângulo.
- vector< PointT > **subCenters**
  Subcentros para divisão.
- float **length**
- float **height**
  Dimensões do retângulo.

4.23.1 Detailed Description

Estrutura que representa um retângulo para tabelas.

4.23.2 Constructor & Destructor Documentation

4.23.2.1 RectangleT() [1/3]

```
RectangleT::RectangleT (
            PointT center,
            float length,
            float height)
```

Construtor com centro, comprimento e altura.

Parameters

center	Centro do retângulo.
length	Comprimento.
height	Altura.

4.23.2.2 RectangleT() [2/3]

```
RectangleT::RectangleT (
            PointT topLeft,
            PointT bottomRight)
```

Construtor com dois pontos.

Parameters

topLeft	Canto superior esquerdo.
bottomRight	Canto inferior direito.

4.23.2.3 RectangleT() [3/3]

```
RectangleT::RectangleT (
            const RectangleT & other)
```

Construtor de cópia.

Parameters

other	Outro retângulo.

4.23.3 Member Function Documentation

4.23.3.1 operator=()

```
RectangleT & RectangleT::operator= (
            const RectangleT & other)
```

Operador de atribuição.

Parameters

other	Outro retângulo.

Returns

Referência para este retângulo.

The documentation for this struct was generated from the following files:

- include/table.hpp
- src/table.cpp

4.24 Register Class Reference

Classe responsável pelo registro de novos perfis.

```
#include <register.hpp>
```

Public Member Functions

- Register (int bufferSize, RectangleT plan)

  Construtor do registro.
- Register (string tittle, int bufferSize, RectangleT plan)

  Construtor do registro com título.
- ~**Register** ()

  Destrutor do registro.
- Register (const Register &other)

  Construtor de cópia.
- Register & operator= (const Register &other)

  Operador de atribuição.
- bool writeInBuffer (char c)

  Escreve um caractere no buffer.
- bool deleteInBuffer ()

  Remove o último caractere do buffer.
- bool cleanBuffer ()

  Limpa o buffer.
- string getTittleContent ()

  Retorna o conteúdo do título.
- string getMessageContent ()

  Retorna o conteúdo da mensagem.
- string getBufferContent ()

  Retorna o conteúdo do buffer.
- string getIthContent (int i)

  Retorna o conteúdo da i-ésima linha.
- Color getTittleTextColor ()

  Retorna a cor do texto do título.
- Color getMessageTextColor ()

  Retorna a cor do texto da mensagem.
- Color getBufferTextColor ()

  Retorna a cor do texto do buffer.
- Color getIthTextColor (int i)

  Retorna a cor do texto da i-ésima linha.
- float getIthCenterX (int i)

  Retorna a coordenada X central da i-ésima linha.
- float getIthCenterY (int i)

  Retorna a coordenada Y central da i-ésima linha.
- void setTittleContent (string tittle)

  Define o conteúdo do título.
- void setMessageContent (string message)

  Define o conteúdo da mensagem.
- void setTittleTextColor (Color color)

  Define a cor do texto do título.
- void setMessageTextColor (Color color)

  Define a cor do texto da mensagem.
- void setBufferTextColor (Color color)

  Define a cor do texto do buffer.
- void drawRegister (ALLEGRO_FONT *font1, ALLEGRO_FONT *font2)

  Desenha o formulário de registro na tela.

4.24.1 Detailed Description

Classe responsável pelo registro de novos perfis.

4.24.2 Constructor & Destructor Documentation

4.24.2.1 Register() [1/3]

```
Register::Register (
            int bufferSize,
            RectangleT plan)
```

Construtor do registro.

Parameters

bufferSize	Tamanho do buffer.
plan	Retângulo de exibição.

4.24.2.2 Register() [2/3]

```
Register::Register (
            string tittle,
            int bufferSize,
            RectangleT plan)
```

Construtor do registro com título.

Parameters

tittle	Título do registro.
bufferSize	Tamanho do buffer.
plan	Retângulo de exibição.

4.24.2.3 Register() [3/3]

```
Register::Register (
            const Register & other)
```

Construtor de cópia.

Parameters

other	Outro objeto Register.

4.24.3 Member Function Documentation

4.24.3.1 cleanBuffer()

```
bool Register::cleanBuffer ()
```

Limpa o buffer.

Returns

true se limpo com sucesso.

4.24.3.2 deleteInBuffer()

```
bool Register::deleteInBuffer ()
```

Remove o último caractere do buffer.

Returns

true se removido com sucesso.

4.24.3.3 drawRegister()

```
void Register::drawRegister (
            ALLEGRO_FONT * font1,
            ALLEGRO_FONT * font2)
```

Desenha o formulário de registro na tela.

Parameters

font1	Fonte principal.
font2	Fonte secundária.

4.24.3.4 getBufferContent()

```
string Register::getBufferContent ()
```

Retorna o conteúdo do buffer.

Returns

Conteúdo do buffer.

4.24.3.5 getBufferTextColor()

```
Color Register::getBufferTextColor ()
```

Retorna a cor do texto do buffer.

Returns

Cor do texto.

4.24.3.6 getIthCenterX()

```
float Register::getIthCenterX (
            int i)
```

Retorna a coordenada X central da i-ésima linha.

Parameters

i	Índice da linha.

Returns

Coordenada X.

4.24.3.7 getIthCenterY()

```
float Register::getIthCenterY (
            int i)
```

Retorna a coordenada Y central da i-ésima linha.

Parameters

i	Índice da linha.

Returns

Coordenada Y.

4.24.3.8 getIthContent()

```
string Register::getIthContent (
            int i)
```

Retorna o conteúdo da i-ésima linha.

Parameters

i	Índice da linha.

Returns

Conteúdo da linha.

4.24.3.9 getIthTextColor()

```
Color Register::getIthTextColor (
            int i)
```

Retorna a cor do texto da i-ésima linha.

Parameters

i	Índice da linha.

Returns

Cor do texto.

4.24.3.10 getMessageContent()

```
string Register::getMessageContent ()
```

Retorna o conteúdo da mensagem.

Returns

Conteúdo da mensagem.

4.24.3.11 getMessageTextColor()

```
Color Register::getMessageTextColor ()
```

Retorna a cor do texto da mensagem.

Returns

Cor do texto.

4.24.3.12 getTittleContent()

```
string Register::getTittleContent ()
```

Retorna o conteúdo do título.

Returns

Conteúdo do título.

4.24.3.13 getTittleTextColor()

```
Color Register::getTittleTextColor ()
```

Retorna a cor do texto do título.

Returns

Cor do texto.

4.24.3.14 operator=()

```
Register & Register::operator= (
            const Register & other)
```

Operador de atribuição.

Parameters

other	Outro objeto Register.

Returns

Referência para este objeto.

4.24.3.15 setBufferTextColor()

```
void Register::setBufferTextColor (
            Color color)
```

Define a cor do texto do buffer.

Parameters

color	Nova cor.

4.24.3.16 setMessageContent()

```
void Register::setMessageContent (
            string message)
```

Define o conteúdo da mensagem.

Parameters

message	Nova mensagem.

4.24.3.17 setMessageTextColor()

```
void Register::setMessageTextColor (
            Color color)
```

Define a cor do texto da mensagem.

Parameters

color	Nova cor.

4.24.3.18 setTittleContent()

```
void Register::setTittleContent (
            string tittle)
```

Define o conteúdo do título.

Parameters

tittle	Novo título.

4.24.3.19 setTittleTextColor()

```
void Register::setTittleTextColor (
            Color color)
```

Define a cor do texto do título.

Parameters

color	Nova cor.

4.24.3.20 writeInBuffer()

```
bool Register::writeInBuffer (
            char c)
```

Escreve um caractere no buffer.

Parameters

c	Caractere a ser escrito.

Returns

true se escrito com sucesso.

The documentation for this class was generated from the following files:

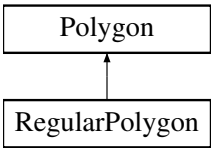
- include/register.hpp
- src/register.cpp

4.25 RegularPolygon Struct Reference

Estrutura que representa um polígono regular (herda de Polygon).

```
#include <polygon.hpp>
```

Inheritance diagram for RegularPolygon:



Public Member Functions

- RegularPolygon (const Point &center, int n, float length)

  Construtor do polígono regular.
- vector< Point > **getVertices** ()

  Retorna os vértices do polígono regular.
- Polygon getPolygon (const Point &center=Point(0, 0)) override

  Retorna o polígono regular.
- void updateVertices (const Point &delta) override

  Atualiza os vértices do polígono regular.

Public Member Functions inherited from Polygon

- vector< Point > getRotatedVertices (const Point &center)

  Retorna os vértices rotacionados em torno de um centro.
- set< Point > **getEdgeVectors** ()

  Retorna o conjunto de vetores das arestas.
- set< Point > **getEdgeNormals** ()

  Retorna o conjunto de vetores normais das arestas.
- **Polygon** (initializer_list< Point > vert)

  Construtor a partir de lista de vértices.
- **Polygon** (const std::vector< Point > &vert)

  Construtor a partir de vetor de vértices.
- **Polygon** (vector< Point > vert, int n, set< Point > vect, set< Point > normals)

  Construtor completo.
- **Polygon** (const Polygon &copy)

  Construtor de cópia.
- void **operator=** (const Polygon &copy)

  Operador de atribuição.
- float * **getPointArray** ()

  Retorna um array de floats com os pontos do polígono.
- void addAngle (float radians)

  Adiciona um ângulo de rotação ao polígono.

Public Attributes

- Point **center**
- int **EdgeCount**
- float **edgeLength**

Public Attributes inherited from Polygon

- vector< Point > **vertices**
- int **vertexCount**
- float **angle** = 0
- set< Point > **edgeVectors**
- set< Point > **edgeNormals**

4.25.1 Detailed Description

Estrutura que representa um polígono regular (herda de Polygon).

4.25.2 Constructor & Destructor Documentation

4.25.2.1 RegularPolygon()

```
RegularPolygon::RegularPolygon (
            const Point & center,
            int n,
            float length)
```

Construtor do polígono regular.

Parameters

center	Centro do polígono.
n	Número de lados.
length	Tamanho da aresta.

4.25.3 Member Function Documentation

4.25.3.1 getPolygon()

```
Polygon RegularPolygon::getPolygon (
            const Point & center = Point(0, 0))  [override], [virtual]
```

Retorna o polígono regular.

Reimplemented from Polygon.

4.25.3.2 updateVertices()

```
void RegularPolygon::updateVertices (
            const Point & delta)  [override], [virtual]
```

Atualiza os vértices do polígono regular.

Reimplemented from Polygon.

The documentation for this struct was generated from the following files:

- include/polygon.hpp
- src/polygon.cpp

4.26 Row Struct Reference

Estrutura que representa uma linha da tabela.

`#include <table.hpp>`

Public Member Functions

- **Row** ()
  Construtor padrão.
- Row (Color textColor, Color rowColor, RectangleT rowRectangle)
  Construtor com cores e retângulo.
- Row (RectangleT rowRectangle)
  Construtor com retângulo.
- Row (PointT topLeft, PointT bottomRight)
  Construtor com dois pontos.
- Row (const Row &other)
  Construtor de cópia.
- Row & operator= (const Row &other)
  Operador de atribuição.
- void **display** ()
  Exibe a linha no console.

Public Attributes

- RectangleT **rowRectangle**
  Retângulo da linha.
- vector< string > **texts**
  Textos da linha.
- Color **textColor**
- Color **rowColor**
  Cores do texto e da linha.

4.26.1 Detailed Description

Estrutura que representa uma linha da tabela.

4.26.2 Constructor & Destructor Documentation

4.26.2.1 Row() [1/4]

```
Row::Row (
            Color textColor,
            Color rowColor,
            RectangleT rowRectangle)
```

Construtor com cores e retângulo.

Parameters

textColor	Cor do texto.
rowColor	Cor da linha.
rowRectangle	Retângulo da linha.

4.26.2.2 Row() [2/4]

```
Row::Row (
            RectangleT rowRectangle)
```

Construtor com retângulo.

Parameters

rowRectangle	Retângulo da linha.

4.26.2.3 Row() [3/4]

```
Row::Row (
            PointT topLeft,
            PointT bottomRight)
```

Construtor com dois pontos.

Parameters

topLeft	Canto superior esquerdo.
bottomRight	Canto inferior direito.

4.26.2.4 Row() [4/4]

```
Row::Row (
            const Row & other)
```

Construtor de cópia.

Parameters

other	Outra linha.

4.26.3 Member Function Documentation

4.26.3.1 operator=()

```
Row & Row::operator= (
            const Row & other)
```

Operador de atribuição.

Parameters

other	Outra linha.

Returns

Referência para esta linha.

The documentation for this struct was generated from the following files:

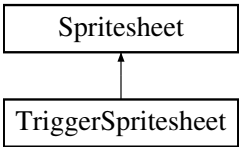
- include/table.hpp
- src/table.cpp

4.27 Spritesheet Class Reference

Classe que representa uma spritesheet para animações.

`#include <animation.hpp>`

Inheritance diagram for Spritesheet:



Public Member Functions

- Spritesheet (const char *dir, int count, int frameW, int frameH, int gap)
  Construtor da spritesheet.
- Spritesheet (const char *dir, int count, int frameW, int gap)
  Construtor alternativo da spritesheet.
- ALLEGRO_BITMAP * getSheet () const
  Retorna o bitmap da spritesheet.
- int getFrameCount () const
  Retorna a quantidade de frames.
- int getFrameWidth () const
  Retorna a largura de cada frame.
- int getFrameHeight () const
  Retorna a altura de cada frame.
- int getCurrentIndex () const
  Retorna o índice do frame atual.
- ALLEGRO_BITMAP * getFrame (int i) const
  Retorna o frame na posição i.
- ALLEGRO_BITMAP * getCurrentFrame () const
  Retorna o frame atual.
- virtual void resetAnimation ()
  Reinicia a animação para o primeiro frame.
- virtual void advanceFrame ()
  Avança para o próximo frame da animação.
- ~**Spritesheet** ()
  Destrutor da spritesheet.

Protected Attributes

- ALLEGRO_BITMAP * **sheet** = nullptr
- int **frameCount**
- int **frameWidth**
- int **frameHeight**
- int **frameGap**
- int **currentIndex** = 0
- vector< ALLEGRO_BITMAP * > **frames**

4.27.1 Detailed Description

Classe que representa uma spritesheet para animações.

4.27.2 Constructor & Destructor Documentation

4.27.2.1 Spritesheet() [1/2]

```
Spritesheet::Spritesheet (
            const char * dir,
            int count,
            int frameW,
            int frameH,
            int gap)
```

Construtor da spritesheet.

Parameters

dir	Caminho do arquivo da spritesheet.
count	Quantidade de frames.
frameW	Largura de cada frame.
frameH	Altura de cada frame.
gap	Espaço entre frames.

4.27.2.2 Spritesheet() [2/2]

```
Spritesheet::Spritesheet (
            const char * dir,
            int count,
            int frameW,
            int gap)
```

Construtor alternativo da spritesheet.

Parameters

dir	Caminho do arquivo da spritesheet.
count	Quantidade de frames.
frameW	Largura de cada frame.
gap	Espaço entre frames.

4.27.3 Member Function Documentation

4.27.3.1 advanceFrame()

```
void Spritesheet::advanceFrame ()  [virtual]
```

Avança para o próximo frame da animação.

Reimplemented in TriggerSpritesheet.

4.27.3.2 getCurrentFrame()

```
ALLEGRO_BITMAP * Spritesheet::getCurrentFrame () const
```

Retorna o frame atual.

Returns

ALLEGRO_BITMAP∗ Frame atual.

4.27.3.3 getCurrentIndex()

```
int Spritesheet::getCurrentIndex () const
```

Retorna o índice do frame atual.

Returns

int Índice do frame atual.

4.27.3.4 getFrame()

```
ALLEGRO_BITMAP * Spritesheet::getFrame (
            int i) const
```

Retorna o frame na posição i.

Parameters

i	Índice do frame.

Returns

ALLEGRO_BITMAP∗ Frame selecionado.

4.27.3.5 getFrameCount()

```
int Spritesheet::getFrameCount () const
```

Retorna a quantidade de frames.

Returns

int Número de frames.

4.27.3.6 getFrameHeight()

```
int Spritesheet::getFrameHeight () const
```

Retorna a altura de cada frame.

Returns

int Altura do frame.

4.27.3.7 getFrameWidth()

```
int Spritesheet::getFrameWidth () const
```

Retorna a largura de cada frame.

Returns

int Largura do frame.

4.27.3.8 getSheet()

```
ALLEGRO_BITMAP * Spritesheet::getSheet () const
```

Retorna o bitmap da spritesheet.

Returns

ALLEGRO_BITMAP∗ Bitmap da spritesheet.

4.27.3.9 resetAnimation()

```
void Spritesheet::resetAnimation ()  [virtual]
```

Reinicia a animação para o primeiro frame.

Reimplemented in TriggerSpritesheet.

The documentation for this class was generated from the following files:

- include/animation.hpp
- src/animation.cpp

4.28 Table Struct Reference

Estrutura que representa uma tabela.

`#include <table.hpp>`

Public Member Functions

- **Table** ()
  Construtor padrão.
- Table (RectangleT tableRectangle)
  Construtor com retângulo.
- Table (const Table &other)
  Construtor de cópia.
- Table & operator= (const Table &other)
  Operador de atribuição.
- void **display** ()
  Exibe a tabela no console.

Public Attributes

- RectangleT **tableRectangle**
  Retângulo da tabela.
- vector< Row > **row**
  Linhas da tabela.

4.28.1 Detailed Description

Estrutura que representa uma tabela.

4.28.2 Constructor & Destructor Documentation

4.28.2.1 Table() [1/2]

```
Table::Table (
            RectangleT tableRectangle)
```

Construtor com retângulo.

Parameters

tableRectangle	Retângulo da tabela.

4.28.2.2 Table() [2/2]

```
Table::Table (
            const Table & other)
```

Construtor de cópia.

Parameters

other	Outra tabela.

4.28.3 Member Function Documentation

4.28.3.1 operator=()

```
Table & Table::operator= (
            const Table & other)
```

Operador de atribuição.

Parameters

other	Outra tabela.

Returns

Referência para esta tabela.

The documentation for this struct was generated from the following files:

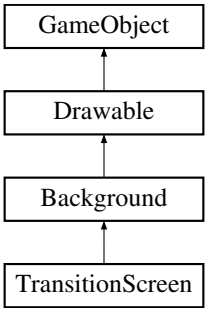
- include/table.hpp
- src/table.cpp

4.29 TransitionScreen Class Reference

Classe que representa a tela de transição do jogo.

```
#include <passive.hpp>
```

Inheritance diagram for TransitionScreen:



Public Member Functions

- **TransitionScreen** ()

  Construtor da tela de transição.
- void updateSpeed () override

  Atualiza a velocidade da transição.
- bool updatePosition () override

  Atualiza a posição da transição.
- void draw () override

  Desenha a tela de transição.
- void **updateStage** ()

  Atualiza o estágio da transição.
- void **startTransition** ()

  Inicia a transição.
- bool isActive ()

  Verifica se a transição está ativa.
- tStage getStage ()

  Retorna o estágio atual da transição.

Public Member Functions inherited from Background

- Background (const char *dir, const Point &pos, float w, float h, float speedX)

  Construtor do plano de fundo.
- virtual ~**Background** ()

  Destrutor virtual do plano de fundo.

Public Member Functions inherited from Drawable

- void addSpeedVector (const Point &spd)

  Adiciona um vetor à velocidade atual.
- void addSpeedX (float x)

  Adiciona um valor à velocidade X.
- void addSpeedY (float y)

  Adiciona um valor à velocidade Y.
- Point getSpeed ()

  Retorna o vetor de velocidade.
- float getSpeedX ()

  Retorna a velocidade no eixo X.
- float getSpeedY ()

  Retorna a velocidade no eixo Y.

Public Member Functions inherited from GameObject

- Point getPos ()

  Retorna a posição do objeto.
- float getPosX ()

  Retorna a coordenada X do objeto.
- float getPosY ()

  Retorna a coordenada Y do objeto.

Additional Inherited Members

Protected Member Functions inherited from Drawable

- Drawable (const Point &pos, const Point &spd)

  Construtor protegido com posição e velocidade.
- Drawable (const Point &pos)

  Construtor protegido apenas com posição.
- void setSpeedX (float x)

  Define a velocidade no eixo X.
- void setSpeedY (float y)

  Define a velocidade no eixo Y.
- void setSpeed (const Point &pos)

  Define o vetor de velocidade.

Protected Member Functions inherited from GameObject

- GameObject (const Point &position)

  Construtor protegido com posição.
- GameObject (float pX, float pY)

  Construtor protegido com coordenadas.
- void setPosX (float x)

  Define a coordenada X do objeto.
- void setPosY (float y)

  Define a coordenada Y do objeto.
- void setPos (const Point &p)

  Define a posição do objeto.

Protected Attributes inherited from Background

- ALLEGRO_BITMAP * **image** = nullptr

  Bitmap do plano de fundo.
- float **width**

  Largura do plano de fundo.
- float **height**

  Altura do plano de fundo.

4.29.1 Detailed Description

Classe que representa a tela de transição do jogo.

4.29.2 Member Function Documentation

4.29.2.1 draw()

```
void TransitionScreen::draw ()  [override], [virtual]
```

Desenha a tela de transição.

Reimplemented from Background.

4.29.2.2 getStage()

```
tStage TransitionScreen::getStage ()
```

Retorna o estágio atual da transição.

Returns

Estágio da transição.

4.29.2.3 isActive()

```
bool TransitionScreen::isActive ()
```

Verifica se a transição está ativa.

Returns

true se ativa.

4.29.2.4 updatePosition()

```
bool TransitionScreen::updatePosition ()  [override], [virtual]
```

Atualiza a posição da transição.

Returns

true se a posição foi atualizada.

Reimplemented from Background.

4.29.2.5 updateSpeed()

```
void TransitionScreen::updateSpeed ()  [override], [virtual]
```

Atualiza a velocidade da transição.

Reimplemented from Background.

The documentation for this class was generated from the following files:

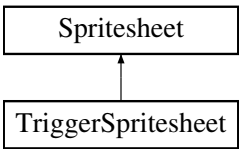
- include/passive.hpp
- src/passive.cpp

4.30 TriggerSpritesheet Class Reference

Classe para spritesheets que são ativadas por gatilho e possuem ciclos.

`#include <animation.hpp>`

Inheritance diagram for TriggerSpritesheet:



Public Member Functions

- TriggerSpritesheet (const char ∗dir, int count, int frameW, int frameH, int gap)
  Construtor da TriggerSpritesheet.
- TriggerSpritesheet (const char ∗dir, int count, int frameW, int gap)
  Construtor alternativo da TriggerSpritesheet.
- void setCycleCount (int n)
  Define o número de ciclos da animação.
- int getCycleCount () const
  Retorna o número de ciclos definidos.
- void resetAnimation () override
  Reinicia a animação e os ciclos.
- void advanceFrame () override
  Avança para o próximo frame, considerando os ciclos.
- bool isActive () const
  Verifica se a animação está ativa.

Public Member Functions inherited from Spritesheet

- Spritesheet (const char ∗dir, int count, int frameW, int frameH, int gap)
  Construtor da spritesheet.
- Spritesheet (const char ∗dir, int count, int frameW, int gap)
  Construtor alternativo da spritesheet.
- ALLEGRO_BITMAP ∗ getSheet () const
  Retorna o bitmap da spritesheet.
- int getFrameCount () const
  Retorna a quantidade de frames.
- int getFrameWidth () const
  Retorna a largura de cada frame.
- int getFrameHeight () const
  Retorna a altura de cada frame.
- int getCurrentIndex () const
  Retorna o índice do frame atual.
- ALLEGRO_BITMAP ∗ getFrame (int i) const
  Retorna o frame na posição i.
- ALLEGRO_BITMAP ∗ getCurrentFrame () const
  Retorna o frame atual.
- ∼**Spritesheet** ()
  Destrutor da spritesheet.

Additional Inherited Members

Protected Attributes inherited from Spritesheet

- ALLEGRO_BITMAP * **sheet** = nullptr
- int **frameCount**
- int **frameWidth**
- int **frameHeight**
- int **frameGap**
- int **currentIndex** = 0
- vector< ALLEGRO_BITMAP * > **frames**

4.30.1 Detailed Description

Classe para spritesheets que são ativadas por gatilho e possuem ciclos.

4.30.2 Constructor & Destructor Documentation

4.30.2.1 TriggerSpritesheet() [1/2]

```
TriggerSpritesheet::TriggerSpritesheet (
            const char * dir,
            int count,
            int frameW,
            int frameH,
            int gap)
```

Construtor da TriggerSpritesheet.

Parameters

dir	Caminho do arquivo da spritesheet.
count	Quantidade de frames.
frameW	Largura de cada frame.
frameH	Altura de cada frame.
gap	Espaço entre frames.

4.30.2.2 TriggerSpritesheet() [2/2]

```
TriggerSpritesheet::TriggerSpritesheet (
            const char * dir,
            int count,
            int frameW,
            int gap)
```

Construtor alternativo da TriggerSpritesheet.

Parameters

dir	Caminho do arquivo da spritesheet.
count	Quantidade de frames.
frameW	Largura de cada frame.
gap	Espaço entre frames.

4.30.3 Member Function Documentation

4.30.3.1 advanceFrame()

```
void TriggerSpritesheet::advanceFrame ()  [override], [virtual]
```

Avança para o próximo frame, considerando os ciclos.

Reimplemented from Spritesheet.

4.30.3.2 getCycleCount()

```
int TriggerSpritesheet::getCycleCount () const
```

Retorna o número de ciclos definidos.

Returns

int Número de ciclos.

4.30.3.3 isActive()

```
bool TriggerSpritesheet::isActive () const
```

Verifica se a animação está ativa.

Returns

true Se ativa.

false Se inativa.

4.30.3.4 resetAnimation()

```
void TriggerSpritesheet::resetAnimation ()  [override], [virtual]
```

Reinicia a animação e os ciclos.

Reimplemented from Spritesheet.

4.30.3.5 setCycleCount()

```
void TriggerSpritesheet::setCycleCount (
            int n)
```

Define o número de ciclos da animação.

Parameters

n	Número de ciclos.

The documentation for this class was generated from the following files:

- include/animation.hpp
- src/animation.cpp

Chapter 5

File Documentation

5.1 animation.hpp

```
00001 #ifndef ANIMATION_H
00002 #define ANIMATION_H
00003
00004 #include <vector>
00005 #include <iostream>
00006
00007 #include <allegro5/allegro.h>
00008
00009 using namespace std;
00010
00012 const float ANIM_FPS = 24;
00013
00017 class Spritesheet
00018 {
00019 protected:
00020     ALLEGRO_BITMAP *sheet = nullptr;
00021     int frameCount, frameWidth, frameHeight, frameGap;
00022     int currentIndex = 0;
00023     vector<ALLEGRO_BITMAP *> frames;
00024
00025 public:
00034     Spritesheet(const char *dir, int count, int frameW, int frameH, int gap);
00035
00043     Spritesheet(const char *dir, int count, int frameW, int gap);
00044
00045     // void loadBitmap(const char* dir);
00046
00051     ALLEGRO_BITMAP *getSheet() const;
00052
00057     int getFrameCount() const;
00058
00063     int getFrameWidth() const;
00064
00069     int getFrameHeight() const;
00070
00075     int getCurrentIndex() const;
00076
00082     ALLEGRO_BITMAP *getFrame(int i) const;
00083
00088     ALLEGRO_BITMAP *getCurrentFrame() const;
00089
00093     virtual void resetAnimation();
00094
00098     virtual void advanceFrame();
00099
00103     ~Spritesheet();
00104 };
00105
00109 class TriggerSpritesheet : public Spritesheet
00110 {
00111 private:
00112     bool active = false;
00113     int cycles = 1;
00114     int currentCycle = 0;
00115
00116 public:
00125     TriggerSpritesheet(const char *dir, int count, int frameW, int frameH, int gap);
```

```
00126
00134     TriggerSpritesheet(const char *dir, int count, int frameW, int gap);
00135
00140     void setCycleCount(int n);
00141
00146     int getCycleCount() const;
00147
00151     void resetAnimation() override;
00152
00156     void advanceFrame() override;
00157
00163     bool isActive() const;
00164 };
00165
00166 #endif
```

5.2 base.hpp

```
00001 #ifndef BASE_H
00002 #define BASE_H
00003 #include <set>
00004 #include <vector>
00005 #include <string>
00006 #include <stdlib.h>
00007 #include <iostream>
00008 #include <fstream>
00009 #include <sstream>
00010 #include <algorithm>
00011
00013 const int NUMPROFILES = 7;
00014
00015 using namespace std;
00016
00020 class Profile
00021 {
00022 private:
00023     string name;
00024     string nickname;
00025     int maxDistance;
00026     int plays;
00027
00028 public:
00032     Profile();
00033
00039     Profile(string, string);
00040
00048     Profile(string, string, int, int);
00049
00054     Profile(const Profile &);
00055
00061     bool operator>(const Profile &) const;
00062
00068     bool operator<(const Profile &) const;
00069
00075     bool operator==(const Profile &) const;
00076
00082     Profile operator=(const Profile &);
00083
00085     string getName();
00086
00088     string getNickname();
00089
00091     int getPlays();
00092
00094     int getMaxDistance();
00095
00097     void setPlays(int);
00098
00100     void setMaxDistance(int);
00101
00103     void display();
00104 };
00105
00109 class Base
00110 {
00111 private:
00112     vector<Profile *> profiles;
00113
00117     void clearProfiles();
00118
00123     void copyProfiles(const vector<Profile *> &);
00124
00125 public:
```

```
00130     Base(const Base &);
00131
00136     Base(const vector<Profile *> &);
00137
00141     Base();
00142
00147     Base(string);
00148
00152     ~Base();
00153
00159     Base operator=(const Base &);
00160
00166     bool inBase(string);
00167
00173     bool inBase(Profile);
00174
00180     bool updateProfiles(Profile);
00181
00187     bool removeProfile(string);
00188
00193     vector<Profile *> getBestProfiles();
00194
00199     void saveBase(string);
00200
00204     void display();
00205 };
00206
00212 bool validateNicknameChars(string);
00213
00219 bool validateNameChars(string);
00220
00226 bool validateNicknameSize(string);
00227
00233 bool validateNameSize(string);
00234
00235 #endif
```

5.3 cooldown.hpp

```
00001 #ifndef COOLDOWN_H
00002 #define COOLDOWN_H
00003
00006 const float FPS = 30;
00007
00011 class Cooldown
00012 {
00013 private:
00014     float timeLeft = 0;
00015     float rechargeTime;
00016     float updateFrequency = 1;
00017     bool frozen = false;
00018
00019 public:
00024     Cooldown(float wuTime);
00025
00029     Cooldown();
00030
00034     void updateCooldown();
00035
00039     void restartCooldown();
00040
00044     void refreshCooldown();
00045
00050     bool isCooldownUp();
00051
00056     void setRechargeTime(float wuTime);
00057
00062     void setUpdateFrequency(float f);
00063
00068     float getCurrentTimeLeft();
00069
00074     float getCurrentPorcentage();
00075
00080     float getRechargeTime();
00081
00085     void freezeCooldown();
00086
00090     void unfreezeCooldown();
00091 };
00092
00093 #endif
```

5.4 entity.hpp

```
00001 #ifndef ENTITY_H
00002 #define ENTITY_H
00003
00004 #include "game_object.hpp"
00005 #include "hitbox.hpp"
00006 #include "animation.hpp"
00007 #include "cooldown.hpp"
00008 #include <string>
00009
00011 const float BASE_GRAVITY = 10;
00013 const float BASE_X_MOVEMENT = 20;
00015 const float X_AXIS = 160;
00016
00020 class Entity : public Drawable
00021 {
00022 protected:
00023     Hitbox *hb;
00024 public:
00030     Entity(const Point &pos, const Point &spd);
00031
00036     Entity(const Point &pos);
00037
00042     Hitbox *getHitbox();
00043
00047     virtual ~Entity();
00048 };
00049
00051 const float PLAYER_SIZE = 20;
00052
00056 enum class PlayerState
00057 {
00058     NONE,
00059     DEAD,
00060     IDLE,
00061     JUMPING
00062 };
00063
00067 class Player : public Entity
00068 {
00069 private:
00070     Spritesheet idleSprite;
00071     TriggerSpritesheet jumpSprite;
00072     PlayerState state = PlayerState::NONE;
00073     Cooldown angleCD;
00074     float angle;
00075 public:
00079     Player();
00080
00085     bool updatePosition() override;
00086
00090     void updateSpeed() override;
00091
00095     void jump();
00096
00100     void draw() override;
00101
00102     // bool loadSprite(const char* dir);
00103
00107     void updatePlayerState();
00108
00112     void updateAnimation();
00113
00118     void setPlayerState(PlayerState s);
00119
00123     ~Player();
00124 };
00125
00127 const float PIPE_X_SPEED = -10;
00128
00132 class Pipe : public Entity
00133 {
00134 private:
00135     ALLEGRO_BITMAP *pipeSprite = nullptr;
00136     bool isInverted;
00137 public:
00138     static float screenSpeed;
00143     static void updateScreenSpeed(float s);
00144
00153     Pipe(const Point &pos, float w, float h, ALLEGRO_BITMAP *image = nullptr, bool inv = false);
00154
00159     bool updatePosition() override;
00160
00164     void updateSpeed() override;
00165
```

```
00170     bool loadSprite();
00171
00175     void draw() override;
00176 };
00177
00179 const float EEL_W = 366;
00181 const float EEL_H = 100;
00182
00186 class Eel : public Pipe
00187 {
00188 private:
00189     Spritesheet *sprite;
00190 public:
00196     Eel(const Point &pos, Spritesheet *image);
00197
00201     void rotate();
00202
00207     bool updatePosition() override;
00208
00212     void draw() override;
00213     //~Eel();
00214 };
00215
00216 #endif
```

5.5 game_object.hpp

```
00001 #ifndef GAME_OBJECT_H
00002 #define GAME_OBJECT_H
00003
00004 #include <allegro5/allegro.h>
00005 #include <allegro5/allegro_primitives.h>
00006 #include "polygon.hpp"
00007
00011 class GameObject
00012 {
00013 private:
00014     Point pos;
00015
00016 protected:
00021     GameObject(const Point &position);
00022
00028     GameObject(float pX, float pY);
00029
00034     void setPosX(float x);
00035
00040     void setPosY(float y);
00041
00046     void setPos(const Point &p);
00047
00048 public:
00053     Point getPos();
00054
00059     float getPosX();
00060
00065     float getPosY();
00066
00071     virtual bool updatePosition() = 0;
00072 };
00073
00077 class Drawable : public GameObject
00078 {
00079 private:
00080     Point speed;
00081 protected:
00087     Drawable(const Point &pos, const Point &spd);
00088
00093     Drawable(const Point &pos);
00094
00099     void setSpeedX(float x);
00100
00105     void setSpeedY(float y);
00106
00111     void setSpeed(const Point &pos);
00112
00113 public:
00117     virtual void draw() = 0;
00118
00122     virtual void updateSpeed() = 0;
00123
00128     void addSpeedVector(const Point &spd);
00129
00134     void addSpeedX(float x);
```

```
00135
00140     void addSpeedY(float y);
00141
00146     Point getSpeed();
00147
00152     float getSpeedX();
00153
00158     float getSpeedY();
00159 };
00160
00161 #endif
```

5.6 game_object_handler.hpp

```
00001 #ifndef HANDLER_H
00002 #define HANDLER_H
00003
00004 #include <list>
00005 #include <memory>
00006 #include <allegro5/allegro.h>
00007 #include "game_object.hpp"
00008 #include "hitbox.hpp"
00009 #include "entity.hpp"
00010
00011 enum Ambient
00012 {
00013     NONE,
00014     FLAPPY,
00015     EELS
00016 };
00017
00018 class Handler
00019 {
00020 private:
00021     bool playing = false;
00022     Player guy;
00023     std::list<std::unique_ptr<Pipe» obstacles;
00024     int time = 0;
00025     Ambient dynamic = NONE;
00026     float gameSpeed = 1;
00027
00028 public:
00039     int gameOn(ALLEGRO_TIMER &timer, ALLEGRO_TIMER &animation_timer, ALLEGRO_EVENT_QUEUE &eventQueue,
      const int SCREEN_H, const int SCREEN_W);
00040
00047     void addObstacle(ALLEGRO_BITMAP *image, Spritesheet *eelImage);
00048
00055     bool outOfBorders();
00056
00063     bool checkCollisions();
00064
00068     void drawAll();
00069
00073     void death();
00074
00082     int sortBetween(int x, int y);
00083
00087     void updateAmbient();
00088
00089     void drawObstacles();
00090 };
00091
00092 #endif
```

5.7 hitbox.hpp

```
00001 #ifndef HITBOX_H
00002 #define HITBOX_H
00003
00004 #include "game_object.hpp"
00005
00011 class Hitbox : public GameObject
00012 {
00013 private:
00014     Drawable *target;
00015 public:
00020     Hitbox(const Point &center);
00021
```

```
00026     bool updatePosition() override;
00027
00032     void setTarget(Drawable *target);
00033
00038     virtual Polygon getPolygon() = 0;
00039
00044     virtual float getAngle() = 0;
00045
00050     virtual void rotateHitbox(float radians) = 0;
00051
00055     virtual ~Hitbox();
00056 };
00057
00061 class RectangleHitbox : public Hitbox
00062 {
00063 private:
00064     float width;
00065     float height;
00066     Rectangle rectangle;
00067 public:
00074     RectangleHitbox(const Point &center, float w, float h);
00075
00076     // bool checkColision() override;
00077
00082     float getWidth();
00083
00088     float getHeight();
00089
00094     Polygon getPolygon() override;
00095
00100     float getAngle() override;
00101
00106     bool updatePosition() override;
00107
00112     void rotateHitbox(float radians) override;
00113
00118     float *getVertices();
00119 };
00120
00124 class PolygonHitbox : public Hitbox
00125 {
00126 private:
00127     int sides;
00128     RegularPolygon polygon;
00129 public:
00136     PolygonHitbox(const Point &center, int n, float EdgeLength);
00137
00142     int getSideCount();
00143
00148     float getEdgeLength();
00149
00154     Polygon getPolygon() override;
00155
00160     float getAngle() override;
00161
00166     bool updatePosition() override;
00167
00172     void rotateHitbox(float radians) override;
00173
00178     float *getVertices();
00179 };
00180
00181 #endif
```

5.8 initializer_allegro.hpp

```
00001 #include <iostream>
00002 #include <allegro5/allegro.h>
00003 #include <allegro5/allegro_primitives.h>
00004 #include <allegro5/allegro_image.h>
00005 #include <allegro5/allegro_font.h>
00006 #include <allegro5/allegro_ttf.h>
00007 #include <allegro5/allegro_audio.h>
00008 #include <allegro5/allegro_acodec.h>
00009
00010 using namespace std;
00011
00018 bool initialize_allegro()
00019 {
00020     // Initialize main allegro
00021     if (!al_init())
00022     {
00023         cout « "ERROR:" « "failed to initialize allegro" « endl;
```

```
00024         return false;
00025     }
00026
00027     // Initialize Allegro primitives addon
00028     if (!al_init_primitives_addon())
00029     {
00030         cout « "ERROR:" « "failed to initialize allegro primitives" « endl;
00031         return false;
00032     }
00033
00034     // Install keyboard input support
00035     if (!al_install_keyboard())
00036     {
00037         cout « "ERROR:" « "failed to initialize keyboard" « endl;
00038         return false;
00039     }
00040     if (!al_init_image_addon())
00041     {
00042         cout « "ERROR:" « "failed to initialize allegro image" « endl;
00043         return false;
00044     }
00045
00046     // install mouse support
00047     if (!al_install_mouse())
00048     {
00049         return false;
00050     }
00051
00052     if (!al_init_font_addon())
00053     {
00054         return false;
00055     }
00056
00057     if (!al_init_ttf_addon())
00058     {
00059         return false;
00060     }
00061
00062     if (!al_install_audio())
00063         return false;
00064
00065     if (!al_init_acodec_addon())
00066         return false;
00067
00068     return true;
00069 }
00070
00078 bool initialize_event_queue(ALLEGRO_EVENT_QUEUE *&ev)
00079 {
00080     ev = al_create_event_queue();
00081     if (!ev)
00082     {
00083         cout « "ERROR:" « "failed to create event_queue" « endl;
00084         return false;
00085     }
00086     else
00087         return true;
00088 }
00089
00101 bool initialize_display_and_timer(ALLEGRO_DISPLAY *&display, int w, int h, ALLEGRO_TIMER *&t, float
      fps)
00102 {
00103     display = al_create_display(w, h);
00104     if (!display)
00105     {
00106         cout « "ERROR:" « "failed to create display" « endl;
00107         return false;
00108     }
00109     else
00110     {
00111         t = al_create_timer(1.0 / fps);
00112         if (!t)
00113         {
00114             cout « "ERROR:" « "failed to initialize timer" « endl;
00115             al_destroy_display(display);
00116             return false;
00117         }
00118         else
00119             return true;
00120     }
00121 }
00122
00132 bool initialize_display(ALLEGRO_DISPLAY *&display, int w, int h)
00133 {
00134     display = al_create_display(w, h);
00135     if (!display)
00136     {
```

```
00137         cout « "ERROR:" « "failed to create display" « endl;
00138         return false;
00139     }
00140     else
00141         return true;
00142 }
00143
00152 bool initialize_timer(ALLEGRO_TIMER *&t, float fps)
00153 {
00154     t = al_create_timer(1.0 / fps);
00155     if (!t)
00156     {
00157         cout « "ERROR:" « "failed to initialize timer" « endl;
00158         return false;
00159     }
00160     else
00161         return true;
00162 }
```

5.9 interface.hpp

```
00001 #include <iostream>
00002 #include <allegro5/allegro.h>
00003 #include <allegro5/allegro_primitives.h>
00004 #include <allegro5/allegro_image.h>
00005 #include <allegro5/allegro_font.h>
00006 #include <allegro5/allegro_ttf.h>
00007 #include <allegro5/allegro_audio.h>
00008 #include <allegro5/allegro_acodec.h>
00009
00010 using namespace std;
00011
00022 bool hover_bool(ALLEGRO_EVENT event, ALLEGRO_BITMAP *any, int xcord, int ycord)
00023 {
00024     int mousex = -1, mousey = -1;
00025     mousex = event.mouse.x;
00026     mousey = event.mouse.y;
00027
00028     int any_w = al_get_bitmap_width(any);
00029     int any_h = al_get_bitmap_height(any);
00030
00031     return (mousex >= xcord && mousex <= xcord + any_w && mousey >= ycord && mousey <= ycord + any_h);
00032 }
```

5.10 leaderboard.hpp

```
00001 #ifndef LEADERBOARD_H
00002 #define LEADERBOARD_H
00003 #include <allegro5/allegro.h>
00004 #include <allegro5/allegro_font.h>
00005 #include <allegro5/allegro_ttf.h>
00006 #include <set>
00007 #include <vector>
00008 #include <string>
00009 #include <iostream>
00010 #include <fstream>
00011 #include <sstream>
00012 #include <algorithm>
00013 #include "base.hpp"
00014 #include "table.hpp"
00015
00019 class LeaderBoard
00020 {
00021 private:
00022     Base base;
00023     vector<Profile *> topProfiles;
00024
00025 public:
00026     Table table;
00027
00033     LeaderBoard(string filename, RectangleT rect);
00034
00040     bool addNewProfile(Profile profile);
00041
00045     void updateLeaderBoard();
00046
00047     // void setTitleRowColor(Color);
00048
```

```
00053     void setFirstRowColor(Color color);
00054
00059     void setSecondRowColor(Color color);
00060
00065     void setThirdRowColor(Color color);
00066
00071     void setOthersRowsColor(Color color);
00072
00073     // void setTitleRowTextColor(Color);
00074
00079     void setFirstRowTextColor(Color color);
00080
00085     void setSecondRowTextColor(Color color);
00086
00091     void setThirdRowTextColor(Color color);
00092
00097     void setOthersRowsTextColor(Color color);
00098
00103     void drawLeaderBoard(ALLEGRO_FONT *font);
00104
00108     void display();
00109
00114     void save(string filename);
00115 };
00116
00117 #endif
```

5.11 passive.hpp

```
00001 #ifndef PASSIVE_H
00002 #define PASSIVE_H
00003
00004 #include "game_object.hpp"
00005 #include "cooldown.hpp"
00006 #include <vector>
00007
00011 class Background : public Drawable
00012 {
00013 protected:
00014     ALLEGRO_BITMAP *image = nullptr;
00015     float width;
00016     float height;
00017 public:
00026     Background(const char *dir, const Point &pos, float w, float h, float speedX);
00027
00031     virtual ~Background();
00032
00036     void updateSpeed() override;
00037
00042     bool updatePosition() override;
00043
00047     void draw() override;
00048
00049     friend class BackgroundHandler;
00050 };
00051
00055 class BackgroundHandler
00056 {
00057 private:
00058     std::vector<Background *> bgPair;
00059     float screenWidth;
00060     float screenHeight;
00061     Point anchor;
00062 public:
00072     BackgroundHandler(const char *dir, float w, float h,
00073                       float speedX, float screenW, float screenH);
00074
00078     ~BackgroundHandler();
00079
00083     void drawBackground();
00084
00088     void updateBackgroundPosition();
00089 };
00090
00094 enum class tStage
00095 {
00096     NONE,
00097     FIRST_HALF,
00098     SECOND_HALF
00099 };
00100
00102 const Point T_ANCHOR(400, 950);
00104 const float T_SPEED = 18;
```

```
00106 const float T_TIME = (750 / (T_SPEED * FPS));
00107
00111 class TransitionScreen : public Background
00112 {
00113 private:
00114     Cooldown cd;
00115     tStage stage = tStage::NONE;
00116 public:
00120     TransitionScreen();
00121
00125     void updateSpeed() override;
00126
00131     bool updatePosition() override;
00132
00136     void draw() override;
00137
00141     void updateStage();
00142
00146     void startTransition();
00147
00152     bool isActive();
00153
00158     tStage getStage();
00159 };
00160
00161 #endif
```

5.12 polygon.hpp

```
00001 #ifndef POLYGON_H
00002 #define POLYGON_H
00003 #include <cmath>
00004 #include <vector>
00005 #include <set>
00006 #include <iostream>
00007
00008 using namespace std;
00009
00011 const float PI = acos(-1.0); // PI calculated with arccos(-1)
00012
00013 // Vector/Point struct, contains a pair of two float coordinates
00017 struct Point
00018 {
00019     float x, y;
00020
00026     Point(float x, float y) : x(x), y(y) {}
00027
00031     Point() : x(0), y(0) {}
00032
00036     Point operator+(const Point &p2) const
00037     {
00038         return Point(x + p2.x, y + p2.y);
00039     }
00040
00044     Point operator-(const Point &p2) const
00045     {
00046         return Point(x - p2.x, y - p2.y);
00047     }
00048
00052     bool operator<(const Point &p2) const
00053     {
00054         return (x != p2.x) ? x < p2.x : y < p2.y;
00055     }
00056
00060     static Point getEdgeVector(const Point &p1, const Point &p2)
00061     {
00062         return p2 - p1;
00063     }
00064
00068     Point getNormalVector() const
00069     {
00070         return Point(-y, x);
00071     }
00072
00076     float dotProduct(const Point &p2) const
00077     {
00078         return x * p2.x + y * p2.y;
00079     }
00080
00084     Point normalizeVector() const
00085     {
00086         double len = std::sqrt(x * x + y * y);
00087         return (len != 0) ? Point(x / len, y / len) : Point(0, 0);
```

```
00088     }
00089
00095     Point rotateVector(float angle) const;
00096
00100     Point rotateVector(float cosA, float sinA) const;
00101
00107     Point rotatePoint(const Point &rotationCenter, float angle) const;
00108
00112     Point rotatePoint(const Point &rotationCenter, float cosA, float sinA) const;
00113 };
00114
00118 ostream &operator«(std::ostream &os, const Point &p);
00119
00120 // Polygon defined by a set of vertices that represent it's size and shape
00124 struct Polygon
00125 {
00126     vector<Point> vertices;
00127     int vertexCount;
00128     float angle = 0;
00129     set<Point> edgeVectors;
00130     set<Point> edgeNormals;
00131
00132     // these functions take into account possible polygon rotation
00137     vector<Point> getRotatedVertices(const Point &center);
00138
00142     set<Point> getEdgeVectors();
00143
00147     set<Point> getEdgeNormals();
00148
00149     // get possibly modified polygon
00154     virtual Polygon getPolygon(const Point &center = Point(0, 0));
00155
00159     Polygon(initializer_list<Point> vert) : vertices(vert)
00160     {
00161         vertexCount = vertices.size();
00162
00163         for (int i = 0; i < vertexCount; i++)
00164         {
00165             Point nextVertex = vertices[(i + 1) % vertexCount];
00166             Point edge = Point::getEdgeVector(vertices[i], nextVertex);
00167
00168             edgeVectors.insert(edge);
00169             edgeNormals.insert(edge.getNormalVector().normalizeVector());
00170         }
00171     }
00172
00176     Polygon(const std::vector<Point> &vert) : vertices(vert)
00177     {
00178         vertexCount = vertices.size();
00179
00180         for (int i = 0; i < vertexCount; i++)
00181         {
00182             Point nextVertex = vertices[(i + 1) % vertexCount];
00183             Point edge = Point::getEdgeVector(vertices[i], nextVertex);
00184
00185             edgeVectors.insert(edge);
00186             edgeNormals.insert(edge.getNormalVector().normalizeVector());
00187         }
00188     }
00189
00193     Polygon(vector<Point> vert, int n, set<Point> vect, set<Point> normals) : vertices(vert),
      vertexCount(n), angle(0), edgeVectors(vect), edgeNormals(normals) {}
00194
00198     Polygon(const Polygon &copy) : vertices(copy.vertices), vertexCount(copy.vertexCount),
00199                                    angle(copy.angle), edgeVectors(copy.edgeVectors),
00200                                    edgeNormals(copy.edgeNormals) {}
00201
00205     void operator=(const Polygon &copy)
00206     {
00207         this->vertices = copy.vertices;
00208         this->vertexCount = copy.vertexCount;
00209         this->angle = copy.angle;
00210         this->edgeVectors = copy.edgeVectors;
00211         this->edgeNormals = copy.edgeNormals;
00212     }
00213
00217     float *getPointArray();
00218
00223     void addAngle(float radians);
00224
00229     virtual void updateVertices(const Point &delta);
00230 };
00231
00235 ostream &operator«(std::ostream &os, const Polygon &p);
00236
00237 // for rouding errors in calculations
00239 const float EPSILON = 1e-5;
```

```
00240
00248 bool isAlmostEqual(float a, float b, float epsilon = EPSILON);
00249
00250 // Used in SAT colision detection
00254 struct PolygonProjection
00255 {
00256     float minProj;
00257     float maxProj;
00258
00264     PolygonProjection(const Polygon &poly, const Point &projAxis)
00265     {
00266         // Point normAxis = projAxis.normalizeVector();
00267
00268         minProj = poly.vertices[0].dotProduct(projAxis);
00269         maxProj = minProj;
00270
00271         for (int i = 1; i < poly.vertexCount; i++)
00272         {
00273             float testedProj = poly.vertices[i].dotProduct(projAxis);
00274
00275             if (testedProj < minProj)
00276                 minProj = testedProj;
00277             if (testedProj > maxProj)
00278                 maxProj = testedProj;
00279         }
00280     }
00281
00287     bool doProjectionOverlap(const PolygonProjection &other)
00288     {
00289         return !(maxProj < other.minProj || other.maxProj < minProj);
00290     }
00291 };
00292
00293 // checks if two polygons colides using Separated Axes Theorem
00294 // remember to always pass as parameters p.getPolygon()!!!!!!!!!!!! (handles rotation right)
00301 bool isColidingSAT(const Polygon &a, const Polygon &b);
00302
00306 struct Rectangle : Polygon
00307 {
00308     float width, height;
00309     Point center;
00310
00317     Rectangle(const Point &center, float w, float h);
00318
00319     // autmatically gets vertices even if rotated in some way, centered in center
00320     // Use normal getRotatedVertices() if rotation center != polygon center of mass
00324     vector<Point> getVertices();
00325
00329     Polygon getPolygon(const Point &center = Point(0, 0)) override;
00330
00334     void updateVertices(const Point &delta) override;
00335 };
00336
00340 struct RegularPolygon : Polygon
00341 {
00342     Point center;
00343     int EdgeCount;
00344     float edgeLength;
00345
00352     RegularPolygon(const Point &center, int n, float length);
00353
00354     // see notes in Rectangle struct
00358     vector<Point> getVertices();
00359
00363     Polygon getPolygon(const Point &center = Point(0, 0)) override;
00364
00368     void updateVertices(const Point &delta) override;
00369 };
00370
00378 std::vector<Point> calculateRectangle(const Point &center, float w, float h);
00379
00387 std::vector<Point> calculateRegularPolygon(const Point &center, int n, float edge);
00388
00394 float *vectorToFloatArray(const std::vector<Point> &points);
00395
00396 #endif
```

5.13 register.hpp

```
00001 #ifndef REGISTER_H
00002 #define REGISTER_H
00003 #include <vector>
00004 #include <stdbool.h>
```

```
00005 #include <string>
00006 #include <vector>
00007 #include <cmath>
00008 #include <iostream>
00009 #include "table.hpp"
00010 #include <set>
00011 #include <allegro5/allegro.h>
00012 #include <allegro5/allegro_primitives.h>
00013 #include <allegro5/allegro_image.h>
00014 #include <allegro5/allegro_font.h>
00015 #include <allegro5/allegro_ttf.h>
00016 #include <allegro5/allegro_audio.h>
00017 #include <allegro5/allegro_acodec.h>
00018
00020 const int MINPROFILENAMESIZE = 4;
00022 const int MAXPROFILENAMESIZE = 20;
00023
00025 const int MINPROFILENICKNAMESIZE = 4;
00027 const int MAXPROFILENICKNAMESIZE = 12;
00028
00030 int const NUMREGISTERROWS = 3;
00031
00035 class Register
00036 {
00037 private:
00038     int bufferSize;
00039     char *buffer;
00040     int index;
00041     string tittle;
00042     string message;
00043     RectangleT plan;
00044     vector<Row> rows;
00045
00046 public:
00052     Register(int bufferSize, RectangleT plan);
00053
00060     Register(string tittle, int bufferSize, RectangleT plan);
00061
00065     ~Register();
00066
00071     Register(const Register &other);
00072
00078     Register &operator=(const Register &other);
00079
00085     bool writeInBuffer(char c);
00086
00091     bool deleteInBuffer();
00092
00097     bool cleanBuffer();
00098
00103     string getTittleContent();
00104
00109     string getMessageContent();
00110
00115     string getBufferContent();
00116
00122     string getIthContent(int i);
00123
00128     Color getTittleTextColor();
00129
00134     Color getMessageTextColor();
00135
00140     Color getBufferTextColor();
00141
00147     Color getIthTextColor(int i);
00148
00154     float getIthCenterX(int i);
00155
00161     float getIthCenterY(int i);
00162
00167     void setTittleContent(string tittle);
00168
00173     void setMessageContent(string message);
00174
00179     void setTittleTextColor(Color color);
00180
00185     void setMessageTextColor(Color color);
00186
00191     void setBufferTextColor(Color color);
00192
00198     void drawRegister(ALLEGRO_FONT *font1, ALLEGRO_FONT *font2);
00199 };
00200
00206 bool validateNameChars(string name);
00207
00213 bool validateNicknameChars(string nickname);
00214
```

```
00220 bool validateNameSize(string name);
00221
00227 bool validateNicknameSize(string nickname);
00228
00234 bool validateName(string name);
00235
00241 bool validateNickname(string nickname);
00242
00249 bool checkName(string name, string &msg);
00250
00257 bool checkNickname(string nickname, string &msg);
00258
00259 #endif
```

5.14 sound.hpp

```
00001 #ifndef ALLINTERFACE_H
00002 #define ALLINTERFACE_H
00003
00004 #include <allegro5/allegro.h>
00005 #include <allegro5/allegro_primitives.h>
00006 #include <allegro5/allegro_image.h>
00007 #include <allegro5/allegro_font.h>
00008 #include <allegro5/allegro_ttf.h>
00009 #include <allegro5/allegro_audio.h>
00010 #include <allegro5/allegro_acodec.h>
00011
00018 void startmusic(ALLEGRO_SAMPLE_INSTANCE *any, float volume)
00019 {
00020     al_attach_sample_instance_to_mixer(any, al_get_default_mixer());
00021     al_set_sample_instance_playmode(any, ALLEGRO_PLAYMODE_LOOP);
00022     al_set_sample_instance_gain(any, volume);
00023 }
00024
00025 #endif
```

5.15 table.hpp

```
00001 #ifndef TABLE_H
00002 #define TABLE_H
00003 #include <vector>
00004 #include <stdbool.h>
00005 #include <string>
00006 #include <vector>
00007 #include <cmath>
00008 #include <iostream>
00009
00010 using namespace std;
00011
00013 const int NUMCOLUMNS = 3;
00015 const int NUMROWS = 7;
00016
00020 struct PointT
00021 {
00022 private:
00028     float toZero(float v);
00029
00030 public:
00031     float x, y;
00032
00036     PointT();
00037
00043     PointT(float x, float y);
00044
00049     PointT(const PointT &other);
00050
00056     PointT &operator=(const PointT &other);
00057
00063     PointT operator-(const PointT &other) const;
00064
00070     PointT operator+(const PointT &other) const;
00071
00077     PointT operator/(float f) const;
00078
00084     PointT operator*(float f);
00085
00089     void display();
00090 };
```

```
00091
00095 struct Color
00096 {
00097     float r, g, b;
00098
00102     Color();
00103
00110     Color(float r, float g, float b);
00111
00116     Color(const Color &other);
00117
00123     Color &operator=(const Color &other);
00124
00130     Color operator/(float f) const;
00131
00135     void display();
00136 };
00137
00141 struct RectangleT
00142 {
00143     PointT center, topLeft, bottomRight;
00144     vector<PointT> subCenters;
00145     float length, height;
00146
00150     RectangleT();
00151
00158     RectangleT(PointT center, float length, float height);
00159
00165     RectangleT(PointT topLeft, PointT bottomRight);
00166
00171     RectangleT(const RectangleT &other);
00172
00178     RectangleT &operator=(const RectangleT &other);
00179
00183     void display();
00184 };
00185
00189 struct Row
00190 {
00191     RectangleT rowRectangle;
00192     vector<string> texts;
00193     Color textColor, rowColor;
00194
00198     Row();
00199
00206     Row(Color textColor, Color rowColor, RectangleT rowRectangle);
00207
00212     Row(RectangleT rowRectangle);
00213
00219     Row(PointT topLeft, PointT bottomRight);
00220
00225     Row(const Row &other);
00226
00232     Row &operator=(const Row &other);
00233
00237     void display();
00238 };
00239
00243 struct Table
00244 {
00245     RectangleT tableRectangle;
00246     vector<Row> row;
00247
00251     Table();
00252
00257     Table(RectangleT tableRectangle);
00258
00263     Table(const Table &other);
00264
00270     Table &operator=(const Table &other);
00271
00275     void display();
00276 };
00277
00278 #endif
```

Index