Zappy GUI

Generated by Doxygen 1.9.6

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 Gui::Errors::AError Class Reference

Base class for custom error types. This class is derived from the `IError` interface and provides a common base for custom error types. It contains a protected member `_message` to store the error message.

`#include <AError.hpp>`

Inheritance diagram for Gui::Errors::AError:

Collaboration diagram for Gui::Errors::AError:

Public Member Functions

- ∼**AError** () override=default

  Destructor.
- const char ∗ what () const noexcept override

  Returns the error message.

Public Member Functions inherited from Gui::Errors::IError

- virtual ∼**IError** ()=default

  Destructor for IError.
- virtual const char ∗ what () const noexcept=0

  Returns a C-style string describing the error. This function must be implemented by derived classes to provide a description of the error. The returned string should be null-terminated.

Protected Attributes

- std::string _message

4.1.1 Detailed Description

Base class for custom error types. This class is derived from the `IError` interface and provides a common base for custom error types. It contains a protected member `_message` to store the error message.

4.1.2 Member Function Documentation

4.1.2.1 what()

```
const char * Gui::Errors::AError::what ( ) const  [override], [virtual], [noexcept]
```

Returns the error message.

Returns

A pointer to a constant character string representing the error message.

Implements Gui::Errors::IError.

4.1.3 Member Data Documentation

4.1.3.1 _message

```
std::string Gui::Errors::AError::_message  [protected]
```

The error message.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/Error/AError.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/Error/Error.cpp

4.2 Gui::AHud Class Reference

Inheritance diagram for Gui::AHud:

4.3 Gui::Decoration Class Reference

Public Member Functions

- Decoration ()

  Construct a new Decoration object.
- ~Decoration ()=default

  Destroy the Decoration object.
- void display (std::pair< std::size_t, std::size_t > mapSize, size_t renderDistance, std::pair< std::size_t, std::size_t > camPos)

  Display decorations.
- Map< bool > getGenerationItem (std::size_t ratio)

  Generate random emplacement for decorations.

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Decoration()

```
Gui::Decoration::Decoration ( )
```

Construct a new Decoration object.

4.3.1.2 ~Decoration()

```
Gui::Decoration::~Decoration ( )  [default]
```

Destroy the Decoration object.

4.3.2 Member Function Documentation

4.3.2.1 display()

```
void Gui::Decoration::display (
            std::pair< std::size_t, std::size_t > mapSize,
            size_t renderDistance,
            std::pair< std::size_t, std::size_t > camPos )
```

Display decorations.

Parameters

mapSize	Size of the map.
renderDistance	Distance to render.
camPos	Position of the camera.

4.3.2.2 getGenerationItem()

```
Map< bool > Gui::Decoration::getGenerationItem (
            std::size_t ratio )
```

Generate random emplacement for decorations.

Parameters

ratio	Ratio fo random emplacement. If ratio = 10 for a tree, there will be one chance at a ten to have a tree on the tile.

Returns

Map<bool> - Boolean list to display item.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/Render/Decoration.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/Render/Decoration.cpp

4.4 Gui::Egg Class Reference

Public Types

- enum **EggState** { **IDLE** , **DEAD** , **BORN** , **HATCHING** }

Public Member Functions

- Egg (size_t id, const std::string &team, std::pair< std::size_t, std::size_t > position)
  Construct a new Egg object.
- ~Egg ()
  Destroy the Egg object.
- std::size_t getId () const
  Get the Id object.
- std::string getTeam () const
  Get the Team object.
- std::pair< std::size_t, std::size_t > getPosition () const
  Get the Position object.
- void setId (std::size_t id)
  Set the id object.
- void setTeam (const std::string &team)
  Set the team object.
- void setPosition (std::pair< std::size_t, std::size_t > position)
  Set the position object.
- void setState (EggState state)
  Set the state object.
- EggState getState () const
  Get the state object.

4.4.1 Constructor & Destructor Documentation

4.4.1.1 Egg()

```
Gui::Egg::Egg (
            size_t id,
            const std::string & team,
            std::pair< std::size_t, std::size_t > position )
```

Construct a new Egg object.

Parameters

id	Id of the egg.
team	Team name of the egg.
position	Position of the egg.

Note

The egg is created when a player lays an egg.

The constructor starts the egg animation if implemented.

4.4.1.2 ~Egg()

```
Gui::Egg::~Egg ( )
```

Destroy the Egg object.

Note

The destructor starts the egg animation if implemented.

4.4.2 Member Function Documentation

4.4.2.1 getId()

```
std::size_t Gui::Egg::getId ( ) const
```

Get the Id object.

Returns

std::size_t Id of the egg.

4.4.2.2 getPosition()

```
std::pair< std::size_t, std::size_t > Gui::Egg::getPosition ( ) const
```

Get the Position object.

Returns

std::pair<std::size_t, std::size_t> Position of the egg.

4.4.2.3 getState()

```
Gui::Egg::EggState Gui::Egg::getState ( ) const
```

Get the state object.

Returns

EggState State of the egg.

4.4.2.4 getTeam()

```
std::string Gui::Egg::getTeam ( ) const
```

Get the Team object.

Returns

std::string Team name of the egg.

4.4.2.5 setId()

```
void Gui::Egg::setId (
            std::size_t id )
```

Set the id object.

Parameters

id	Id of the egg.

4.4.2.6 setPosition()

```
void Gui::Egg::setPosition (
            std::pair< std::size_t, std::size_t > position )
```

Set the position object.

Parameters

position	Position of the egg.

4.4.2.7 setState()

```
void Gui::Egg::setState (
            EggState state )
```

Set the state object.

Parameters

state	State of the egg.

4.4.2.8 setTeam()

```
void Gui::Egg::setTeam (
            const std::string & team )
```

Set the team object.

Parameters

team	Team name of the egg.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/GameDatas/Egg.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/GameDatas/Egg.cpp

4.5 Gui::Engine Class Reference

Public Member Functions

- Engine (std::shared_ptr< Network > network)

  Construct a new Engine object.
- ~Engine ()=default

  Destroy the Engine object.
- void run (void)

  Run the engine loop.

4.5.1 Constructor & Destructor Documentation

4.5.1.1 Engine()

```
Gui::Engine::Engine (
            std::shared_ptr< Network > network )
```

Construct a new Engine object.

Parameters

network	Network class.

4.5.1.2 ~Engine()

```
Gui::Engine::~Engine ( )  [default]
```

Destroy the Engine object.

4.5.2 Member Function Documentation

4.5.2.1 run()

```
void Gui::Engine::run (
            void  )
```

Run the engine loop.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/Engine/Engine.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/Engine/Engine.cpp

4.6 Gui::Errors::Error Class Reference

Base class for argument-related errors.

```
#include <Error.hpp>
```

Inheritance diagram for Gui::Errors::Error:

Collaboration diagram for Gui::Errors::Error:

Additional Inherited Members

Public Member Functions inherited from Gui::Errors::AError

- ~**AError** () override=default

  Destructor.
- const char * what () const noexcept override

  Returns the error message.

Public Member Functions inherited from Gui::Errors::IError

- virtual ~**IError** ()=default

  Destructor for IError.
- virtual const char * what () const noexcept=0

  Returns a C-style string describing the error. This function must be implemented by derived classes to provide a description of the error. The returned string should be null-terminated.

Protected Attributes inherited from Gui::Errors::AError

- std::string _message

4.6.1 Detailed Description

Base class for argument-related errors.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/Error/Error.hpp

4.7 Gui::Event Class Reference

Public Member Functions

- Event ()

  Construct a new Event object.
- ~Event ()=default

  Destroy the Event object.
- void listen ()

  Listen the user's events.
- void setRender (std::shared_ptr< Render > render)

  Set the Render object.
- void setGameData (std::shared_ptr< GameData > gameData)

  Set the GameData object.

4.7.1 Constructor & Destructor Documentation

4.7.1.1 Event()

```
Gui::Event::Event ( )
```

Construct a new Event object.

4.7.1.2 ~Event()

```
Gui::Event::~Event ( )  [default]
```

Destroy the Event object.

4.7.2 Member Function Documentation

4.7.2.1 listen()

```
void Gui::Event::listen ( )
```

Listen the user's events.

4.7.2.2 setGameData()

```
void Gui::Event::setGameData (
            std::shared_ptr< GameData > gameData )
```

Set the GameData object.

Parameters

gameData	GameData class.

4.7.2.3 setRender()

```
void Gui::Event::setRender (
            std::shared_ptr< Render > render )
```

Set the Render object.

Parameters

render	Render class.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/Event/Event.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/Event/Event.cpp

4.8 Gui::GameData Class Reference

Public Types

- enum **TimeUnitState** { **INCREASE** , **DECREASE** , **NONE** }

Public Member Functions

- GameData ()

  Construct a new GameData object.
- ~GameData ()=default

  Destroy the GameData object.
- std::vector< Gui::Team > & getTeams ()

  Get the Teams object.
- Gui::Team & getTeam (const std::string &name)

  Get a Team object.
- void addTeam (const Gui::Team &team)

  Add a team to the game.
- void addTeam (const std::string &name, const std::string &playerModelPath, const std::string &eggModel↩ Path)

  Add a team to the game.
- void addPlayerToTeam (const std::string &teamName, const Gui::Player &player)

  Add a player to a team.
- Gui::Player & getPlayer (size_t id)

  Get a player object.
- Map< Gui::Tile > & getMap ()

  Get the Map object.
- void setMap (const Map< Gui::Tile > &map)

  Set the Map object.
- void setMapSize (size_t x, size_t y)

  Set the Map Size object.
- std::pair< size_t, size_t > getMapSize () const

  Get the Map Size object.
- Gui::Tile & getTile (size_t x, size_t y)

  Get a Tile object.
- void setTile (const Gui::Tile &tile)

  Set the Tile object.
- void restartLastTick (void)

  Restart the last tick clock.

- void setServerTick (std::size_t tick)
  Set the Server Tick object.
- clock_t getLastTick () const
  Get the Last Tick object.
- std::size_t getServerTick () const
  Get the Server Tick object.
- void setIsEndGame (bool isEndGame)
  Set the IsEnd Game object.
- bool getIsEndGame () const
  Get the IsEnd Game object.
- void setLastError (const std::string &error)
  Set the Last Error object.
- std::string getLastError () const
  Get the Last Error object.
- Team & getTeamById (std::size_t id)
  Get the Team From Player object.
- TimeUnitState getTimeUnitFromServer () const
  Get the Time Unit From Server object.
- void setTimeUnitFromServer (TimeUnitState timeUnitFromServer)
  Set the Time Unit From Server object.

4.8.1 Constructor & Destructor Documentation

4.8.1.1 GameData()

```
Gui::GameData::GameData ( )
```

Construct a new GameData object.

4.8.1.2 ~GameData()

```
Gui::GameData::~GameData ( )  [default]
```

Destroy the GameData object.

4.8.2 Member Function Documentation

4.8.2.1 addPlayerToTeam()

```
void Gui::GameData::addPlayerToTeam (
            const std::string & teamName,
            const Gui::Player & player )
```

Add a player to a team.

Parameters

teamName	Name of the team.
player	Player to add.

4.8.2.2 addTeam() [1/2]

```
void Gui::GameData::addTeam (
            const Gui::Team & team )
```

Add a team to the game.

Parameters

team	Team to add.

4.8.2.3 addTeam() [2/2]

```
void Gui::GameData::addTeam (
            const std::string & name,
            const std::string & playerModelPath,
            const std::string & eggModelPath )
```

Add a team to the game.

Parameters

name	Name of the team.
playerModelPath	Path to the asset of the team for players.
eggModelPath	Path to the asset of the team for eggs.

4.8.2.4 getIsEndGame()

```
bool Gui::GameData::getIsEndGame ( ) const
```

Get the IsEnd Game object.

Returns

true - The game is finished.

false - The game continue.

4.8.2.5 getLastError()

```
std::string Gui::GameData::getLastError ( ) const
```

Get the Last Error object.

Returns

std::string - Last error message.

4.8.2.6 getLastTick()

```
clock_t Gui::GameData::getLastTick ( ) const
```

Get the Last Tick object.

Returns

clock_t - Last Tick.

4.8.2.7 getMap()

```
Map< Gui::Tile > & Gui::GameData::getMap ( )
```

Get the Map object.

Returns

Map<Gui::Tile>& Map of the game.

4.8.2.8 getMapSize()

```
std::pair< size_t, size_t > Gui::GameData::getMapSize ( ) const
```

Get the Map Size object.

Returns

std::pair<size_t, size_t> Size of the map.

4.8.2.9 getPlayer()

```
Gui::Player & Gui::GameData::getPlayer (
            size_t id )
```

Get a player object.

Parameters

id	Id of the player.

4.8.2.10 getServerTick()

```
std::size_t Gui::GameData::getServerTick ( ) const
```

Get the Server Tick object.

Returns

std::size_t - Server Tick.

4.8.2.11 getTeam()

```
Gui::Team & Gui::GameData::getTeam (
            const std::string & name )
```

Get a Team object.

Parameters

name	Name of the team.

Returns

Gui::Team& Team object.

4.8.2.12 getTeamById()

```
Gui::Team & Gui::GameData::getTeamById (
            std::size_t id )
```

Get the Team From Player object.

Parameters

id	Id of the player.

Returns

Gui::Team& Team of the player.

4.8.2.13 getTeams()

```
std::vector< Gui::Team > & Gui::GameData::getTeams ( )
```

Get the Teams object.

Returns

std::vector<Gui::Team>& Teams of the game.

4.8.2.14 getTile()

```
Gui::Tile & Gui::GameData::getTile (
            size_t x,
            size_t y )
```

Get a Tile object.

Parameters

x	X position of the tile.
y	Y position of the tile.

Returns

Gui::Tile& Tile object.

4.8.2.15 getTimeUnitFromServer()

```
Gui::GameData::TimeUnitState Gui::GameData::getTimeUnitFromServer ( ) const
```

Get the Time Unit From Server object.

Returns

true - The time unit has changed.

4.8.2.16 restartLastTick()

```
void Gui::GameData::restartLastTick (
            void  )
```

Restart the last tick clock.

4.8.2.17 setIsEndGame()

```
void Gui::GameData::setIsEndGame (
            bool isEndGame )
```

Set the IsEnd Game object.

Parameters

isEndGame	EndGame state.

4.8.2.18 setLastError()

```
void Gui::GameData::setLastError (
            const std::string & error )
```

Set the Last Error object.

Parameters

error	Error message.

4.8.2.19 setMap()

```
void Gui::GameData::setMap (
            const Map< Gui::Tile > & map )
```

Set the Map object.

Parameters

map	Map of the game.

4.8.2.20 setMapSize()

```
void Gui::GameData::setMapSize (
            size_t x,
            size_t y )
```

Set the Map Size object.

Parameters

x	X size of the map.
y	Y size of the map.

Note

This method resizes the map.

4.8.2.21 setServerTick()

```
void Gui::GameData::setServerTick (
            std::size_t tick )
```

Set the Server Tick object.

Parameters

tick	Tick of the server.

4.8.2.22 setTile()

```
void Gui::GameData::setTile (
            const Gui::Tile & tile )
```

Set the Tile object.

Parameters

x	X position of the tile.
y	Y position of the tile.
tile	Tile to set.

4.8.2.23 setTimeUnitFromServer()

```
void Gui::GameData::setTimeUnitFromServer (
            TimeUnitState timeUnitFromServer )
```

Set the Time Unit From Server object.

Parameters

timeUnitFromServer	Time unit state.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/GameDatas/GameData.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/GameDatas/GameData.cpp

4.9 Gui::Errors::GuiGameDataException Class Reference

Error class for GameData errors.

`#include <Error.hpp>`

Inheritance diagram for Gui::Errors::GuiGameDataException:

Collaboration diagram for Gui::Errors::GuiGameDataException:

Public Member Functions

- GuiGameDataException (std::string message)

  Constructor for GuiGameDataException.

Public Member Functions inherited from Gui::Errors::AError

- ~**AError** () override=default

  Destructor.
- const char * what () const noexcept override

  Returns the error message.

Public Member Functions inherited from Gui::Errors::IError

- virtual ~**IError** ()=default

  Destructor for IError.
- virtual const char * what () const noexcept=0

  Returns a C-style string describing the error. This function must be implemented by derived classes to provide a description of the error. The returned string should be null-terminated.

Additional Inherited Members

Protected Attributes inherited from Gui::Errors::AError

- std::string _message

4.9.1 Detailed Description

Error class for GameData errors.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 GuiGameDataException()

```
Gui::Errors::GuiGameDataException::GuiGameDataException (
            std::string message )
```

Constructor for GuiGameDataException.

Parameters

message	The error message.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/Error/Error.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/Error/Error.cpp

4.10 Gui::GUIUpdater Class Reference

Public Member Functions

- GUIUpdater (std::shared_ptr< GameData > gameData, std::shared_ptr< Network > network)
  Construct a new GUIUpdater object.
- ~**GUIUpdater** ()=default
  Destroy the GUIUpdater object.
- void update (const std::string &command, const std::vector< std::string > &data)
  Update the GUI GameData.

4.10.1 Constructor & Destructor Documentation

4.10.1.1 GUIUpdater()

```
Gui::GUIUpdater::GUIUpdater (
            std::shared_ptr< GameData > gameData,
            std::shared_ptr< Network > network )
```

Construct a new GUIUpdater object.

Parameters

gameData	The GUI GameData to update.
network	The network to send commands to the server.

4.10.2 Member Function Documentation

4.10.2.1 update()

```
void Gui::GUIUpdater::update (
            const std::string & command,
            const std::vector< std::string > & data )
```

Update the GUI GameData.

Parameters

command	The command to update the GUI GameData.
data	The data to update the GUI GameData.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/GUIUpdater/GUIUpdater.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/GUIUpdater/GUIUpdater.cpp

4.11 Gui::Errors::GuiUpdaterException Class Reference

Error class for GUIUpdater errors.

```
#include <Error.hpp>
```

Inheritance diagram for Gui::Errors::GuiUpdaterException:

Collaboration diagram for Gui::Errors::GuiUpdaterException:

Public Member Functions

- GuiUpdaterException (std::string message)

  Constructor for GuiUpdaterException.

Public Member Functions inherited from Gui::Errors::AError

- ~**AError** () override=default

  Destructor.
- const char * what () const noexcept override

  Returns the error message.

Public Member Functions inherited from Gui::Errors::IError

- virtual ~**IError** ()=default

  Destructor for IError.
- virtual const char * what () const noexcept=0

  Returns a C-style string describing the error. This function must be implemented by derived classes to provide a description of the error. The returned string should be null-terminated.

Additional Inherited Members

Protected Attributes inherited from Gui::Errors::AError

- std::string _message

4.11.1 Detailed Description

Error class for GUIUpdater errors.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 GuiUpdaterException()

```
Gui::Errors::GuiUpdaterException::GuiUpdaterException (
            std::string message )
```

Constructor for GuiUpdaterException.

Parameters

message	The error message.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/Error/Error.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/Error/Error.cpp

4.12 Gui::HudGame Class Reference

Inheritance diagram for Gui::HudGame:

Collaboration diagram for Gui::HudGame:

Public Member Functions

- HudGame (std::shared_ptr< GameData > gameData)
  Construct a new Hud Game object.
- ~HudGame ()=default
  Destroy the Hud Game object.
- void display ()
  Display Game Hud.

Public Member Functions inherited from Gui::AHud

- ~AHud ()=default
  Destroy the AHud object.
- virtual void display ()=0
  Display Hud.
- void setPlayer (std::shared_ptr< Player > player)
  Set the Player object.
- void setTile (std::shared_ptr< Tile > tile)
  Set the Tile object.
- TypeScene getType () const
  Get the Type object.

Public Member Functions inherited from Gui::IHud

- virtual ~IHud ()=default
  Destroy the IHud object.
- virtual void display ()=0
  Display the Hud.
- virtual void setPlayer (std::shared_ptr< Player > player)=0
  Set the Player object.
- virtual void setTile (std::shared_ptr< Tile > tile)=0
  Set the Tile object.
- virtual TypeScene getType () const =0
  Get the Type object.

Additional Inherited Members

Public Types inherited from Gui::IHud

- enum TypeScene { **GAME** , **POV_PLAYER** , **END_GAME** , **TILE** }

  Hud enum for the different scenes.

Protected Attributes inherited from Gui::AHud

- TypeScene **_typeScene**
- std::shared_ptr< GameData > **_gameData**
- std::shared_ptr< Player > **_player**
- std::shared_ptr< Tile > **_tile**

4.12.1 Constructor & Destructor Documentation

4.12.1.1 HudGame()

```
Gui::HudGame::HudGame (
            std::shared_ptr< GameData > gameData )
```

Construct a new Hud Game object.

Parameters

gameData	GameData class.

4.12.1.2 ~HudGame()

```
Gui::HudGame::~HudGame ( )  [default]
```

Destroy the Hud Game object.

4.12.2 Member Function Documentation

4.12.2.1 display()

```
void Gui::HudGame::display ( )  [virtual]
```

Display Game Hud.

Implements Gui::AHud.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/Hud/HudGame.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/Hud/HudGame.cpp

4.13 Gui::HudPlayer Class Reference

Inheritance diagram for Gui::HudPlayer:

Collaboration diagram for Gui::HudPlayer:

Public Member Functions

- HudPlayer (std::shared_ptr< GameData > gameData)
  Construct a new Hud Player object.
- ~HudPlayer ()=default
  Destroy the Hud Player object.
- void display ()
  Display Player Hud.

Public Member Functions inherited from Gui::AHud

- ~AHud ()=default
  Destroy the AHud object.
- virtual void display ()=0
  Display Hud.
- void setPlayer (std::shared_ptr< Player > player)
  Set the Player object.
- void setTile (std::shared_ptr< Tile > tile)
  Set the Tile object.
- TypeScene getType () const
  Get the Type object.

Public Member Functions inherited from Gui::IHud

- virtual ~IHud ()=default
  Destroy the IHud object.
- virtual void display ()=0
  Display the Hud.
- virtual void setPlayer (std::shared_ptr< Player > player)=0
  Set the Player object.
- virtual void setTile (std::shared_ptr< Tile > tile)=0
  Set the Tile object.
- virtual TypeScene getType () const =0
  Get the Type object.

Additional Inherited Members

Public Types inherited from Gui::IHud

- enum TypeScene { **GAME** , **POV_PLAYER** , **END_GAME** , **TILE** }

  Hud enum for the different scenes.

Protected Attributes inherited from Gui::AHud

- TypeScene **_typeScene**
- std::shared_ptr< GameData > **_gameData**
- std::shared_ptr< Player > **_player**
- std::shared_ptr< Tile > **_tile**

4.13.1 Constructor & Destructor Documentation

4.13.1.1 HudPlayer()

```
Gui::HudPlayer::HudPlayer (
            std::shared_ptr< GameData > gameData )
```

Construct a new Hud Player object.

Parameters

gameData	GameData class.

4.13.1.2 ~HudPlayer()

```
Gui::HudPlayer::~HudPlayer ( )  [default]
```

Destroy the Hud Player object.

4.13.2 Member Function Documentation

4.13.2.1 display()

```
void Gui::HudPlayer::display ( )  [virtual]
```

Display Player Hud.

Implements Gui::AHud.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/Hud/HudPlayer.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/Hud/HudPlayer.cpp

4.14 Gui::HudTile Class Reference

Inheritance diagram for Gui::HudTile:

Collaboration diagram for Gui::HudTile:

Public Member Functions

- HudTile (std::shared_ptr< GameData > gameData)
  Construct a new Hud Tile object.
- ~HudTile ()=default
  Destroy the Hud Tile object.
- void display ()
  Display Tile Hud.

Public Member Functions inherited from Gui::AHud

- ~AHud ()=default
  Destroy the AHud object.
- virtual void display ()=0
  Display Hud.
- void setPlayer (std::shared_ptr< Player > player)
  Set the Player object.
- void setTile (std::shared_ptr< Tile > tile)
  Set the Tile object.
- TypeScene getType () const
  Get the Type object.

Public Member Functions inherited from Gui::IHud

- virtual ~IHud ()=default
  Destroy the IHud object.
- virtual void display ()=0
  Display the Hud.
- virtual void setPlayer (std::shared_ptr< Player > player)=0
  Set the Player object.
- virtual void setTile (std::shared_ptr< Tile > tile)=0
  Set the Tile object.
- virtual TypeScene getType () const =0
  Get the Type object.

Additional Inherited Members

Public Types inherited from Gui::IHud

- enum TypeScene { **GAME** , **POV_PLAYER** , **END_GAME** , **TILE** }

  Hud enum for the different scenes.

Protected Attributes inherited from Gui::AHud

- TypeScene **_typeScene**
- std::shared_ptr< GameData > **_gameData**
- std::shared_ptr< Player > **_player**
- std::shared_ptr< Tile > **_tile**

4.14.1 Constructor & Destructor Documentation

4.14.1.1 HudTile()

```
Gui::HudTile::HudTile (
            std::shared_ptr< GameData > gameData )
```

Construct a new Hud Tile object.

Parameters

gameData	GameData class.

4.14.1.2 ~HudTile()

```
Gui::HudTile::~HudTile ( )  [default]
```

Destroy the Hud Tile object.

4.14.2 Member Function Documentation

4.14.2.1 display()

```
void Gui::HudTile::display ( )  [virtual]
```

Display Tile Hud.

Implements Gui::AHud.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/Hud/HudTile.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/Hud/HudTile.cpp

4.15 Gui::Errors::IError Class Reference

Inheritance diagram for Gui::Errors::IError:

Collaboration diagram for Gui::Errors::IError:

Public Member Functions

- virtual ~**IError** ()=default

  Destructor for IError.
- virtual const char * what () const noexcept=0

  Returns a C-style string describing the error. This function must be implemented by derived classes to provide a description of the error. The returned string should be null-terminated.

4.15.1 Member Function Documentation

4.15.1.1 what()

```
virtual const char * Gui::Errors::IError::what ( ) const  [pure virtual], [noexcept]
```

Returns a C-style string describing the error. This function must be implemented by derived classes to provide a description of the error. The returned string should be null-terminated.

Returns

const char* A C-style string describing the error.

Implemented in Gui::Errors::AError.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/Error/IError.hpp

4.16 Gui::IHud Class Reference

Inheritance diagram for Gui::IHud:

Public Types

- enum TypeScene { **GAME** , **POV_PLAYER** , **END_GAME** , **TILE** }

  Hud enum for the different scenes.

Public Member Functions

- virtual ~IHud ()=default

  Destroy the IHud object.
- virtual void display ()=0

  Display the Hud.
- virtual void setPlayer (std::shared_ptr< Player > player)=0

  Set the Player object.
- virtual void setTile (std::shared_ptr< Tile > tile)=0

  Set the Tile object.
- virtual TypeScene getType () const =0

  Get the Type object.

4.16.1 Member Enumeration Documentation

4.16.1.1 TypeScene

```
enum Gui::IHud::TypeScene
```

Hud enum for the different scenes.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 ~IHud()

```
virtual Gui::IHud::~IHud ( )  [virtual], [default]
```

Destroy the IHud object.

4.16.3 Member Function Documentation

4.16.3.1 display()

```
virtual void Gui::IHud::display ( )  [pure virtual]
```

Display the Hud.

Implemented in Gui::HudGame, Gui::HudPlayer, Gui::HudTile, and Gui::AHud.

4.16.3.2 getType()

```
virtual TypeScene Gui::IHud::getType ( ) const  [pure virtual]
```

Get the Type object.

Returns

TypeScene - Type of the scene.

Implemented in Gui::AHud.

4.16.3.3 setPlayer()

```
virtual void Gui::IHud::setPlayer (
            std::shared_ptr< Player > player )  [pure virtual]
```

Set the Player object.

Parameters

player	Player to display infos.

Implemented in Gui::AHud.

4.16.3.4 setTile()

```
virtual void Gui::IHud::setTile (
            std::shared_ptr< Tile > tile )  [pure virtual]
```

Set the Tile object.

Parameters

tile	Tile to display infos.

Implemented in Gui::AHud.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/Hud/IHud.hpp

4.17 Gui::Inventory Class Reference

Public Types

- using **Ressources** = size_t[RessourcesNumber]

Public Member Functions

- Inventory ()

  Construct a new Inventory object.
- Inventory (std::size_t food, std::size_t linemate, std::size_t deraumere, std::size_t sibur, std::size_t mendiane, std::size_t phiras, std::size_t thystame)

  Construct a new Inventory object.
- ~Inventory ()=default

  Destroy the Inventory object.
- void setFood (std::size_t food)

  Set the Food object.
- void setLinemate (std::size_t linemate)

  Set the Linemate object.
- void setDeraumere (std::size_t deraumere)

  Set the Deraumere object.
- void setSibur (std::size_t sibur)

  Set the Sibur object.
- void setMendiane (std::size_t mendiane)

  Set the Mendiane object.
- void setPhiras (std::size_t phiras)

  Set the Phiras object.
- void setThystame (std::size_t thytsame)

  Set the Thystame object.
- void setRessources (Ressources ressources)

  Set the Ressources object.
- std::size_t getFood (void)

  Get the Food object.
- std::size_t getLinemate (void)

  Get the Linemate object.
- std::size_t getDeraumere (void)

  Get the Deraumere object.
- std::size_t getSibur (void)

  Get the Sibur object.
- std::size_t getMendiane (void)

  Get the Mendiane object.
- std::size_t getPhiras (void)

  Get the Phiras object.
- std::size_t getThystame (void)

  Get the Thystame object.
- Ressources & getRessources (void)

  Get the Ressources object.

4.17.1 Constructor & Destructor Documentation

4.17.1.1 Inventory() [1/2]

```
Gui::Inventory::Inventory ( )
```

Construct a new Inventory object.

4.17.1.2 Inventory() [2/2]

```
Gui::Inventory::Inventory (
            std::size_t food,
            std::size_t linemate,
            std::size_t deraumere,
            std::size_t sibur,
            std::size_t mendiane,
            std::size_t phiras,
            std::size_t thystame )
```

Construct a new Inventory object.

Parameters

food	Food to set.
linemate	Linemate to set.
deraumere	Deraumere to set.
sibur	Sibur to set.
mendiane	Mendiane to set.
phiras	Phiras to set.
thystame	Thystame to set.

4.17.1.3 ~Inventory()

```
Gui::Inventory::~Inventory ( )  [default]
```

Destroy the Inventory object.

4.17.2 Member Function Documentation

4.17.2.1 getDeraumere()

```
std::size_t Gui::Inventory::getDeraumere (
            void  )
```

Get the Deraumere object.

Returns

std::size_t - deraumere

4.17.2.2 getFood()

```
std::size_t Gui::Inventory::getFood (
            void  )
```

Get the Food object.

Returns

std::size_t - food

4.17.2.3 getLinemate()

```
std::size_t Gui::Inventory::getLinemate (
            void  )
```

Get the Linemate object.

Returns

std::size_t - linemate

4.17.2.4 getMendiane()

```
std::size_t Gui::Inventory::getMendiane (
            void  )
```

Get the Mendiane object.

Returns

std::size_t - mendiane

4.17.2.5 getPhiras()

```
std::size_t Gui::Inventory::getPhiras (
            void  )
```

Get the Phiras object.

Returns

std::size_t - phiras

4.17.2.6 getRessources()

```
Gui::Inventory::Ressources & Gui::Inventory::getRessources (
            void  )
```

Get the Ressources object.

Returns

Ressources - ressources

4.17.2.7 getSibur()

```
std::size_t Gui::Inventory::getSibur (
            void  )
```

Get the Sibur object.

Returns

std::size_t - sibur

4.17.2.8 getThystame()

```
std::size_t Gui::Inventory::getThystame (
            void  )
```

Get the Thystame object.

Returns

std::size_t - thystame

4.17.2.9 setDeraumere()

```
void Gui::Inventory::setDeraumere (
            std::size_t deraumere )
```

Set the Deraumere object.

Parameters

deraumere	Deraumere to set.

4.17.2.10 setFood()

```
void Gui::Inventory::setFood (
            std::size_t food )
```

Set the Food object.

Parameters

food	Food to set.

4.17.2.11 setLinemate()

```
void Gui::Inventory::setLinemate (
            std::size_t linemate )
```

Set the Linemate object.

Parameters

linemate	Linemate to set.

4.17.2.12 setMendiane()

```
void Gui::Inventory::setMendiane (
            std::size_t mendiane )
```

Set the Mendiane object.

Parameters

mendiane	Mendiane to set.

4.17.2.13 setPhiras()

```
void Gui::Inventory::setPhiras (
            std::size_t phiras )
```

Set the Phiras object.

Parameters

phiras	Phiras to set.

4.17.2.14 setRessources()

```
void Gui::Inventory::setRessources (
            Ressources ressources )
```

Set the Ressources object.

Parameters

ressources	Ressources to set.

4.17.2.15 setSibur()

```
void Gui::Inventory::setSibur (
            std::size_t sibur )
```

Set the Sibur object.

Parameters

sibur	Sibur to set.

4.17.2.16 setThystame()

```
void Gui::Inventory::setThystame (
            std::size_t thytsame )
```

Set the Thystame object.

Parameters

thystame	Thystame to set.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/GameDatas/Inventory.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/GameDatas/Inventory.cpp

4.18 Gui::Network Class Reference

Public Member Functions

- Network (int port, const std::string &hostName)

  Construct a new Network object.
- ~Network ()=default

  Destroy the Network object.
- void setPort (int port)

  Set the port object.
- void setHostName (const std::string &hostName)

  Set the host name object.
- int getPort () const

  Get the port object.
- std::string getHostName () const

  Get the host name object.
- void connectToServer ()

  Connect the Gui network with the server.
- const std::string listenServer ()

  Listen the server and return it message.
- void sendMessageServer (const std::string &message)

  Send a message to the Server.

4.18.1 Constructor & Destructor Documentation

4.18.1.1 Network()

```
Gui::Network::Network (
            int port,
            const std::string & hostName )
```

Construct a new Network object.

Parameters

port	Port of the server.
hostName	Host of the server.

4.18.1.2 ~Network()

```
Gui::Network::~Network ( )  [default]
```

Destroy the Network object.

4.18.2 Member Function Documentation

4.18.2.1 connectToServer()

```
void Gui::Network::connectToServer ( )
```

Connect the Gui network with the server.

4.18.2.2 getHostName()

```
std::string Gui::Network::getHostName ( ) const
```

Get the host name object.

Returns

const std::string - Host name of the server.

4.18.2.3 getPort()

```
int Gui::Network::getPort ( ) const
```

Get the port object.

Returns

const int - Port of the server.

4.18.2.4 listenServer()

```
const std::string Gui::Network::listenServer ( )
```

Listen the server and return it message.

Returns

std::string - Message of the server.

4.18.2.5 sendMessageServer()

```
void Gui::Network::sendMessageServer (
            const std::string & message )
```

Send a message to the Server.

Parameters

message	Message to send to the server.

4.18.2.6 setHostName()

```
void Gui::Network::setHostName (
            const std::string & hostName )
```

Set the host name object.

Parameters

hostName	Host of the server.

4.18.2.7 setPort()

```
void Gui::Network::setPort (
            int port )
```

Set the port object.

Parameters

port	Port of the server.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/Network/Network.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/Network/Network.cpp

4.19 Gui::Errors::NetworkException Class Reference

Error class for network errors.

```
#include <Error.hpp>
```

Inheritance diagram for Gui::Errors::NetworkException:

Collaboration diagram for Gui::Errors::NetworkException:

Public Member Functions

- NetworkException (std::string message)

  Constructor for NetworkException.

Public Member Functions inherited from Gui::Errors::AError

- ~**AError** () override=default

  Destructor.
- const char * what () const noexcept override

  Returns the error message.

Public Member Functions inherited from Gui::Errors::IError

- virtual ~**IError** ()=default

  Destructor for IError.
- virtual const char * what () const noexcept=0

  Returns a C-style string describing the error. This function must be implemented by derived classes to provide a description of the error. The returned string should be null-terminated.

Additional Inherited Members

Protected Attributes inherited from Gui::Errors::AError

- std::string _message

4.19.1 Detailed Description

Error class for network errors.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 NetworkException()

```
Gui::Errors::NetworkException::NetworkException (
            std::string message )
```

Constructor for NetworkException.

Parameters

message	The error message.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/Error/Error.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/Error/Error.cpp

4.20 Gui::ParseCommandLine Class Reference

Public Member Functions

- ParseCommandLine (int argc, char **argv)

  Construct a new Parse Command Line object.
- ~ParseCommandLine ()=default

  Destroy the Parse Command Line object.
- void parseFlags (int argc, char **argv)

  Parse flags in command line.
- int getPort (void)

  Get the port object.
- std::string getHostName (void)

  Get the hostName object.

4.20.1 Constructor & Destructor Documentation

4.20.1.1 ParseCommandLine()

```
Gui::ParseCommandLine::ParseCommandLine (
            int argc,
            char ** argv )
```

Construct a new Parse Command Line object.

Parameters

argc	Number of arguments in command line.
argv	Array with command line arguments.

4.20.1.2 ~ParseCommandLine()

```
Gui::ParseCommandLine::~ParseCommandLine ( )  [default]
```

Destroy the Parse Command Line object.

4.20.2 Member Function Documentation

4.20.2.1 getHostName()

```
std::string Gui::ParseCommandLine::getHostName (
            void  )
```

Get the hostName object.

Returns

std::string - hostName

4.20.2.2 getPort()

```
int Gui::ParseCommandLine::getPort (
            void  )
```

Get the port object.

Returns

int - port

4.20.2.3 parseFlags()

```
void Gui::ParseCommandLine::parseFlags (
            int argc,
            char ** argv )
```

Parse flags in command line.

Parameters

argc	Number of arguments in command line.
argv	Array with command line arguments.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/Parsing/ParseCommandLine.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/Parsing/ParseCommandLine.cpp

4.21 Gui::Errors::ParseCommandLineException Class Reference

Error class for parseCommandLine errors.

`#include <Error.hpp>`

Inheritance diagram for Gui::Errors::ParseCommandLineException:

Collaboration diagram for Gui::Errors::ParseCommandLineException:

Public Member Functions

- ParseCommandLineException (std::string message)
  Constructor for ParseCommandLineException.

Public Member Functions inherited from Gui::Errors::AError

- ~**AError** () override=default
  Destructor.
- const char * what () const noexcept override
  Returns the error message.

Public Member Functions inherited from Gui::Errors::IError

- virtual ~**IError** ()=default
  Destructor for IError.
- virtual const char * what () const noexcept=0
  Returns a C-style string describing the error. This function must be implemented by derived classes to provide a description of the error. The returned string should be null-terminated.

Additional Inherited Members

Protected Attributes inherited from Gui::Errors::AError

- std::string _message

4.21.1 Detailed Description

Error class for parseCommandLine errors.

4.21.2 Constructor & Destructor Documentation

4.21.2.1 ParseCommandLineException()

```
Gui::Errors::ParseCommandLineException::ParseCommandLineException (
            std::string message )
```

Constructor for ParseCommandLineException.

Parameters

message	The error message.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/Error/Error.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/Error/Error.cpp

4.22 Gui::Player Class Reference

Collaboration diagram for Gui::Player:

Public Types

- enum **PlayerState** {
  IDLE , **BORN** , **BROADCAST** , **EJECT** ,
  BEING_EJECTED , **WALK** , **INCANTATION** , **FINISHED_INCANTATION** ,
  LAY_EGG , **DROP** , **COLLECT** , **DEAD** }

Public Member Functions

- Player (std::size_t id, const std::string &team, std::pair< std::size_t, std::size_t > position, std::size_t orientation, std::size_t level=1)

  Construct a new Player object.
- ~Player ()=default

  Destroy the Player object.
- void setPosition (std::pair< std::size_t, std::size_t > position)

  Set the Position object.
- void setId (std::size_t id)

  Set the Id object.
- void setLevel (std::size_t level)

  Set the Level object.
- void setOrientation (std::size_t orientation)

  Set the Orientation object.
- void setTeam (const std::string &team)

  Set the Team object.
- std::pair< std::size_t, std::size_t > getPosition (void) const

  Get the Position object.
- std::size_t getId (void) const

  Get the Id object.
- std::size_t getLevel (void) const

  Get the Level object.
- std::size_t getOrientation (void) const

  Get the Orientation object.
- std::string getTeam (void) const

  Get the Team object.

- void setState (PlayerState state)

  Set the State object.
- PlayerState getState (void) const

  Get the State object.
- void setBroadcast (const std::string &broadcast)

  Set the Broadcast object.
- std::string getBroadcast () const

  Get the Broadcast object.
- float getRotationFromOrientation () const

  Get the Vector From Orientation object.
- Vector3 getCenterPosition ()

  Get the Center Position object.

Public Attributes

- Inventory inventory

  Inventory of the player.

4.22.1 Constructor & Destructor Documentation

4.22.1.1 Player()

```
Gui::Player::Player (
            std::size_t id,
            const std::string & team,
            std::pair< std::size_t, std::size_t > position,
            std::size_t orientation,
            std::size_t level = 1 )
```

Construct a new Player object.

Parameters

id	Id of the player.
team	Team name of the player.
position	Position of the Player.

4.22.1.2 ~Player()

```
Gui::Player::~Player ( )  [default]
```

Destroy the Player object.

4.22.2 Member Function Documentation

4.22.2.1 getBroadcast()

```
std::string Gui::Player::getBroadcast ( ) const
```

Get the Broadcast object.

Returns

std::string - Broadcast message.

4.22.2.2 getCenterPosition()

```
Vector3 Gui::Player::getCenterPosition ( )
```

Get the Center Position object.

Returns

Vector3 - Center position.

4.22.2.3 getId()

```
std::size_t Gui::Player::getId (
            void  ) const
```

Get the Id object.

Returns

std::size_t - id

4.22.2.4 getLevel()

```
std::size_t Gui::Player::getLevel (
            void  ) const
```

Get the Level object.

Returns

std::size_t - level

4.22.2.5 getOrientation()

```
std::size_t Gui::Player::getOrientation (
            void  ) const
```

Get the Orientation object.

Returns

std::size_t - orientation

4.22.2.6 getPosition()

```
std::pair< std::size_t, std::size_t > Gui::Player::getPosition (
            void  ) const
```

Get the Position object.

Returns

std::pair<std::size_t, std::size_t> - position

4.22.2.7 getRotationFromOrientation()

```
float Gui::Player::getRotationFromOrientation ( ) const
```

Get the Vector From Orientation object.

Returns

Vector3 - Vector3 from orientation.

4.22.2.8 getState()

```
Gui::Player::PlayerState Gui::Player::getState (
            void  ) const
```

Get the State object.

Returns

PlayerState - Player state.

4.22.2.9 getTeam()

```
std::string Gui::Player::getTeam (
            void  ) const
```

Get the Team object.

Returns

std::string - team name

4.22.2.10 setBroadcast()

```
void Gui::Player::setBroadcast (
            const std::string & broadcast )
```

Set the Broadcast object.

Parameters

broadcast	New broadcast message.

4.22.2.11 setId()

```
void Gui::Player::setId (
            std::size_t id )
```

Set the Id object.

Parameters

id	Id of the player.

4.22.2.12 setLevel()

```
void Gui::Player::setLevel (
            std::size_t level )
```

Set the Level object.

Parameters

level	Level of the player.

4.22.2.13 setOrientation()

```
void Gui::Player::setOrientation (
            std::size_t orientation )
```

Set the Orientation object.

Parameters

orientation	Orientation of the player.

4.22.2.14 setPosition()

```
void Gui::Player::setPosition (
            std::pair< std::size_t, std::size_t > position )
```

Set the Position object.

Parameters

position	Position of the player

4.22.2.15 setState()

```
void Gui::Player::setState (
            PlayerState state )
```

Set the State object.

Parameters

state	New player state.

4.22.2.16 setTeam()

```
void Gui::Player::setTeam (
            const std::string & team )
```

Set the Team object.

Parameters

team	Team name of the player.

4.22.3 Member Data Documentation

4.22.3.1 inventory

```
Inventory Gui::Player::inventory
```

Inventory of the player.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/GameDatas/Player.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/GameDatas/Player.cpp

4.23 Gui::Render Class Reference

Public Member Functions

- Render (std::shared_ptr< GameData > gameData)
  Construct a new Render object.
- ~Render ()
  Destroy the Render object.
- bool isOpen ()
  Check if the window is open.
- void draw ()
  Draw the scene.
- std::shared_ptr< Camera > getCamera ()
  Get the Camera object.
- void setIsDebug (bool isDebug)
  Set the Is Debug object.
- bool getIsDebug (void)
  Get the Is Debug object.
- void setCameraType (Gui::UserCamera::CameraType type)
  Set the Type object.
- Gui::UserCamera::CameraType getCameraType () const

Get the Type object.

- void setCameraPlayerPov (std::size_t id)

  Set the Camera player pov id.
- std::size_t getCameraPlayerPov () const

  Get the Camera player pov id.
- void setCameraTile (std::pair< std::size_t, std::size_t > pos)

  Set the Camera Tile object.
- std::pair< std::size_t, std::size_t > getCameraTile () const

  Get the Camera Tile object.
- Model getTileModel () const

  Get the Tile model.
- void setRenderDistance (size_t renderDistance)

  Set the Render Distance value.
- size_t getRenderDistance () const

  Get the Render Distance value.
- bool isCameraInPlayerPov () const

  Check if the camera is in player pov.
- size_t getTimeUnit () const

  Get the Time Unit value.
- void setTimeUnit (size_t timeUnit)

  Set the Time Unit value.

4.23.1 Constructor & Destructor Documentation

4.23.1.1 Render()

```
Gui::Render::Render (
            std::shared_ptr< GameData > gameData )
```

Construct a new Render object.

4.23.1.2 ~Render()

```
Gui::Render::~Render ( )
```

Destroy the Render object.

4.23.2 Member Function Documentation

4.23.2.1 draw()

```
void Gui::Render::draw ( )
```

Draw the scene.

4.23.2.2 getCamera()

```
std::shared_ptr< Camera > Gui::Render::getCamera ( )
```

Get the Camera object.

Returns

std::shared_ptr<Camera> - camera

4.23.2.3 getCameraPlayerPov()

```
std::size_t Gui::Render::getCameraPlayerPov ( ) const
```

Get the Camera player pov id.

Returns

std::size_t - Id of the player.

4.23.2.4 getCameraTile()

```
std::pair< std::size_t, std::size_t > Gui::Render::getCameraTile ( ) const
```

Get the Camera Tile object.

Returns

std::pair<std::size_t, std::size_t> - Tile position.

4.23.2.5 getCameraType()

```
Gui::UserCamera::CameraType Gui::Render::getCameraType ( ) const
```

Get the Type object.

Returns

CameraType - Camera type.

4.23.2.6 getIsDebug()

```
bool Gui::Render::getIsDebug (
            void  )
```

Get the Is Debug object.

Returns

true - diplay debug

false - do not display debug

4.23.2.7 getRenderDistance()

```
size_t Gui::Render::getRenderDistance ( ) const
```

Get the Render Distance value.

4.23.2.8 getTileModel()

```
Model Gui::Render::getTileModel ( ) const
```

Get the Tile model.

4.23.2.9 getTimeUnit()

```
size_t Gui::Render::getTimeUnit ( ) const
```

Get the Time Unit value.

Returns

size_t - Time unit value.

4.23.2.10 isCameraInPlayerPov()

```
bool Gui::Render::isCameraInPlayerPov ( ) const
```

Check if the camera is in player pov.

Returns

true - Camera is in player pov.

false - Camera is not in player pov.

Note

The player pov is the first person, second person and third person.

4.23.2.11 isOpen()

```
bool Gui::Render::isOpen ( )
```

Check if the window is open.

Returns

true - the window is open

false - the window is closed

4.23.2.12 setCameraPlayerPov()

```
void Gui::Render::setCameraPlayerPov (
            std::size_t id )
```

Set the Camera player pov id.

Parameters

id	Id of the player.

4.23.2.13 setCameraTile()

```
void Gui::Render::setCameraTile (
            std::pair< std::size_t, std::size_t > pos )
```

Set the Camera Tile object.

Parameters

pos	Tile position.

4.23.2.14 setCameraType()

```
void Gui::Render::setCameraType (
            Gui::UserCamera::CameraType type )
```

Set the Type object.

Parameters

type	Type to set.

4.23.2.15 setIsDebug()

```
void Gui::Render::setIsDebug (
            bool isDebug )
```

Set the Is Debug object.

Parameters

isDebug	New Is Debug value to set.

4.23.2.16 setRenderDistance()

```
void Gui::Render::setRenderDistance (
            size_t renderDistance )
```

Set the Render Distance value.

Parameters

renderDistance	New render distance value.

4.23.2.17 setTimeUnit()

```
void Gui::Render::setTimeUnit (
            size_t timeUnit )
```

Set the Time Unit value.

Parameters

timeUnit	New time unit value.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/Render/Render.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/Render/Render.cpp

4.24 Gui::ServerParser Class Reference

Public Types

- enum ParseType { **INT** , **STRING** , **MESSAGE** , **LIST_INT** }
  Enum of types to parse.

Public Member Functions

- ServerParser ()=default
  Construct a new Server Parser object.
- ~ServerParser ()=default
  Destroy the Server Parser object.
- std::vector< std::string > parse (const std::string &command)
  Parse the command server.

4.24.1 Member Enumeration Documentation

4.24.1.1 ParseType

```
enum Gui::ServerParser::ParseType
```

Enum of types to parse.

4.24.2 Constructor & Destructor Documentation

4.24.2.1 ServerParser()

```
Gui::ServerParser::ServerParser ( )  [default]
```

Construct a new Server Parser object.

4.24.2.2 ~ServerParser()

```
Gui::ServerParser::~ServerParser ( )  [default]
```

Destroy the Server Parser object.

4.24.3 Member Function Documentation

4.24.3.1 parse()

```
std::vector< std::string > Gui::ServerParser::parse (
            const std::string & command )
```

Parse the command server.

Parameters

command	Command to parse.

Returns

std::vector<std::string> - arguments parsed

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/Parsing/ServerParser.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/Parsing/ServerParser.cpp

4.25 Gui::Errors::ServerParserException Class Reference

Error class for network errors.

```
#include <Error.hpp>
```

Inheritance diagram for Gui::Errors::ServerParserException:

Collaboration diagram for Gui::Errors::ServerParserException:

Public Member Functions

- ServerParserException (std::string message)

  Constructor for ServerParserException.

Public Member Functions inherited from Gui::Errors::AError

- ~**AError** () override=default

  Destructor.
- const char * what () const noexcept override

  Returns the error message.

Public Member Functions inherited from Gui::Errors::IError

- virtual ~**IError** ()=default

  Destructor for IError.
- virtual const char * what () const noexcept=0

  Returns a C-style string describing the error. This function must be implemented by derived classes to provide a description of the error. The returned string should be null-terminated.

Additional Inherited Members

Protected Attributes inherited from Gui::Errors::AError

- std::string _message

4.25.1 Detailed Description

Error class for network errors.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 ServerParserException()

```
Gui::Errors::ServerParserException::ServerParserException (
            std::string message )
```

Constructor for ServerParserException.

Parameters

message	The error message.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/Error/Error.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/Error/Error.cpp

4.26 Gui::Team Class Reference

Public Member Functions

- Team (const std::string &name, const std::string &playerModelPath, const std::string &eggModelPath)

  Construct a new Team object.
- ~Team ()

  Destroy the Team object.
- const std::string & getName () const

  Get the Name object.
- std::vector< Gui::Player > & getPlayers ()

  Get the Players object.
- std::vector< Gui::Egg > & getEggs ()

  Get the Eggs object.
- void setName (const std::string &name)

  Set the Name object.
- void addPlayer (const Gui::Player &player)

  Add a player to the team.
- void addEgg (const Gui::Egg &egg)

  Add an egg to the team.
- bool removePlayer (std::size_t id)

  Remove a player from the team.
- bool removeEgg (std::size_t id)

  Remove an egg from the team.
- std::shared_ptr< Gui::Player > getPlayer (std::size_t id)

  Get the Player object.
- Model getPlayerModel () const

  Get the Model object.
- void setPlayerModelPath (const std::string &playerModelPath)

  Set the Model object.
- std::shared_ptr< Gui::Egg > getEgg (std::size_t id)

  Get the Egg object.
- Model getEggModel () const

  Get the Egg Model Path object.
- void setEggModelPath (const std::string &eggModelPath)

  Set the Egg Model Path object.
- std::vector< BoundingBox > getPlayerBoundingBoxes (std::pair< size_t, size_t > pos, size_t orientation, Vector3 center)

  Get the Player Boundig Boxes object.
- Vector3 getPlayerPositionIn3DSpace (size_t id, Map< Tile > map)

  Get the Player position in 3D space.
- std::vector< RayCollision > getPlayerModelHitbox (size_t id, Camera camera)

  Get the Player Model hitbox.
- bool isPlayerHit (size_t id, Camera camera)

  Check if the player is hit.

4.26.1 Constructor & Destructor Documentation

4.26.1.1 Team()

```
Gui::Team::Team (
            const std::string & name,
            const std::string & playerModelPath,
            const std::string & eggModelPath )
```

Construct a new Team object.

Parameters

name	Name of the team.
playerModelPath	Path to the team model asset for players.
eggSkinPath	Path to the skin of the team.

4.26.1.2 ~Team()

```
Gui::Team::~Team ( )
```

Destroy the Team object.

4.26.2 Member Function Documentation

4.26.2.1 addEgg()

```
void Gui::Team::addEgg (
            const Gui::Egg & egg )
```

Add an egg to the team.

Parameters

egg	Egg to add.

4.26.2.2 addPlayer()

```
void Gui::Team::addPlayer (
            const Gui::Player & player )
```

Add a player to the team.

Parameters

player	Player to add.

4.26.2.3 getEgg()

```
std::shared_ptr< Gui::Egg > Gui::Team::getEgg (
            std::size_t id )
```

Get the Egg object.

Parameters

id	Id of the egg.

Returns

std::shared_ptr<Gui::Egg> Egg.

4.26.2.4 getEggModel()

```
Model Gui::Team::getEggModel ( ) const
```

Get the Egg Model Path object.

Returns

const std::string& Path to the eggs Model of the team.

4.26.2.5 getEggs()

```
std::vector< Gui::Egg > & Gui::Team::getEggs ( )
```

Get the Eggs object.

Returns

std::vector<Gui::Egg>& Eggs of the team.

4.26.2.6 getName()

```
const std::string & Gui::Team::getName ( ) const
```

Get the Name object.

Returns

const std::string& Name of the team.

4.26.2.7 getPlayer()

```
std::shared_ptr< Gui::Player > Gui::Team::getPlayer (
            std::size_t id )
```

Get the Player object.

Parameters

id	Id of the player.

Returns

std::shared_ptr<Gui::Player> Player.

4.26.2.8 getPlayerBoundingBoxes()

```
std::vector< BoundingBox > Gui::Team::getPlayerBoundingBoxes (
            std::pair< size_t, size_t > pos,
            size_t orientation,
            Vector3 center )
```

Get the Player Boundig Boxes object.

Parameters

pos	Position of the player.
orientation	Orientation of the player.

Returns

std::vector<BoundingBox> Bounding boxes of the player.

4.26.2.9 getPlayerModel()

```
Model Gui::Team::getPlayerModel ( ) const
```

Get the Model object.

Returns

Model - Model asset of the Team.

4.26.2.10 getPlayerModelHitbox()

```
std::vector< RayCollision > Gui::Team::getPlayerModelHitbox (
            size_t id,
            Camera camera )
```

Get the Player Model hitbox.

Parameters

id	Id of the player.
camera	Camera of the game.

Returns

std::vector<RayCollision> Hitbox of the player.

4.26.2.11 getPlayerPositionIn3DSpace()

```
Vector3 Gui::Team::getPlayerPositionIn3DSpace (
            size_t id,
            Map< Tile > map )
```

Get the Player position in 3D space.

Parameters

id	Id of the player.
map	Map of the game.

4.26.2.12 getPlayers()

```
std::vector< Gui::Player > & Gui::Team::getPlayers ( )
```

Get the Players object.

Returns

std::vector<Gui::Player>& Players of the team.

4.26.2.13 isPlayerHit()

```
bool Gui::Team::isPlayerHit (
            size_t id,
            Camera camera )
```

Check if the player is hit.

Parameters

id	Id of the player.
camera	Camera of the game.

Returns

true If the player is hit.

4.26.2.14 removeEgg()

```
bool Gui::Team::removeEgg (
            std::size_t id )
```

Remove an egg from the team.

Parameters

id	Id of the egg to remove.

Returns

true If the egg has been removed.

false If the egg has not been removed.

4.26.2.15 removePlayer()

```
bool Gui::Team::removePlayer (
            std::size_t id )
```

Remove a player from the team.

Parameters

id	Id of the player to remove.

Returns

true If the player has been removed.

false If the player has not been removed.

4.26.2.16 setEggModelPath()

```
void Gui::Team::setEggModelPath (
            const std::string & eggModelPath )
```

Set the Egg Model Path object.

Parameters

eggSkinPath	Path to the eggs Model of the team.

4.26.2.17 setName()

```
void Gui::Team::setName (
            const std::string & name )
```

Set the Name object.

Parameters

name	Name of the team.

4.26.2.18 setPlayerModelPath()

```
void Gui::Team::setPlayerModelPath (
            const std::string & playerModelPath )
```

Set the Model object.

Parameters

playerModelPath	Path to the team model asset for players.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/GameDatas/Team.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/GameDatas/Team.cpp

4.27 Gui::Tile Class Reference

Collaboration diagram for Gui::Tile:

Public Member Functions

- Tile (std::pair< std::size_t, std::size_t > position)

  Construct a new Tile object.
- Tile (std::pair< std::size_t, std::size_t > position, Inventory inventory)

  Construct a new Tile object.
- ~Tile ()=default

  Destroy the Tile object.
- void setPosition (std::pair< std::size_t, std::size_t > position)

  Set the Position object.
- std::pair< std::size_t, std::size_t > getPosition () const

  Get the Position object.
- Vector3 getPositionIn3DSpace ()

  Get the Position In Space object.
- std::vector< BoundingBox > getTileBoundingBoxes (Tile tile, Model tileModel)

  Get the Tile Bounding Boxes object.
- std::vector< RayCollision > getTileModelHitbox (Tile tile, Camera camera, Model tileModel)

  Get the Tile Model Hitbox object.
- bool isTileHit (Camera camera, Model _tileModel)

  Check if the tile is hit.

Public Attributes

- Inventory inventory

  Inventory of the tile.

4.27.1 Constructor & Destructor Documentation

4.27.1.1 Tile() [1/2]

```
Gui::Tile::Tile (
            std::pair< std::size_t, std::size_t > position )
```

Construct a new Tile object.

Parameters

position	Set the position of the tile.

4.27.1.2 Tile() [2/2]

```
Gui::Tile::Tile (
            std::pair< std::size_t, std::size_t > position,
            Inventory inventory )
```

Construct a new Tile object.

Parameters

position	Set the position of the tile.
inventory	Set the inventory of the tile.

4.27.1.3 ~Tile()

```
Gui::Tile::~Tile ( )  [default]
```

Destroy the Tile object.

4.27.2 Member Function Documentation

4.27.2.1 getPosition()

```
std::pair< std::size_t, std::size_t > Gui::Tile::getPosition ( ) const
```

Get the Position object.

Returns

std::pair<std::size_t, std::size_t> - position x y

4.27.2.2 getPositionIn3DSpace()

```
Vector3 Gui::Tile::getPositionIn3DSpace ( )
```

Get the Position In Space object.

Returns

Vector3 - Position in space.

4.27.2.3 getTileBoundingBoxes()

```
std::vector< BoundingBox > Gui::Tile::getTileBoundingBoxes (
            Tile tile,
            Model tileModel )
```

Get the Tile Bounding Boxes object.

Parameters

tile	Tile to get the bounding boxes.

Returns

std::vector<BoundingBox> - Bounding boxes of the tile.

4.27.2.4 getTileModelHitbox()

```
std::vector< RayCollision > Gui::Tile::getTileModelHitbox (
            Tile tile,
            Camera camera,
            Model tileModel )
```

Get the Tile Model Hitbox object.

Parameters

tile	Tile to get the hitbox.
camera	Camera to get the hitbox.

Returns

std::vector<RayCollision> - Hitbox of the tile.

4.27.2.5 isTileHit()

```
bool Gui::Tile::isTileHit (
            Camera camera,
            Model _tileModel )
```

Check if the tile is hit.

Parameters

camera	Camera to check if the tile is hit.
_tileModel	Model of the tile.

Returns

true - The tile is hit.

false - The tile is not hit.

4.27.2.6 setPosition()

```
void Gui::Tile::setPosition (
            std::pair< std::size_t, std::size_t > position )
```

Set the Position object.

Parameters

position	New position of the tile.

4.27.3 Member Data Documentation

4.27.3.1 inventory

```
Inventory Gui::Tile::inventory
```

Inventory of the tile.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/GameDatas/Tile.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/GameDatas/Tile.cpp

4.28 Gui::UserCamera Class Reference

Public Types

- enum **CameraType** {
  FREE , **FIRST_PERSON** , **SECOND_PERSON** , **THIRD_PERSON** ,
  FREE_TILE }

Public Member Functions

- UserCamera ()

  Construct a new User Camera object.
- ~UserCamera ()=default

  Destroy the User Camera object.
- void setPosition (Vector3 position)

  Set the Position object.
- void setTarget (Vector3 target)

  Set the Target object.
- void setUp (Vector3 up)

  Set the Up object.
- void setFovy (float fovy)

  Set the Fovy object.
- Vector3 getPosition (void) const

  Get the Position object.
- Vector3 getTarget (void) const

  Get the Target object.
- Vector3 getUp (void) const

  Get the Up object.
- float getFovy (void) const

  Get the Fovy object.
- std::shared_ptr< Camera > getCamera ()

  Get the Camera object.
- void setType (CameraType type)

  Set the Type object.
- CameraType getType () const

  Get the Type object.
- void setPlayerId (size_t playerId)

  Set the Player Id object.
- size_t getPlayerId () const

  Get the Player Id object.
- void setTilePos (std::pair< std::size_t, std::size_t > pos)

  Set the Tile Pos object.
- std::pair< std::size_t, std::size_t > getTilePos () const

  Get the Tile position object.
- bool isPlayerPov () const

  Check if the camera is in player pov.

4.28.1 Constructor & Destructor Documentation

4.28.1.1 UserCamera()

```
Gui::UserCamera::UserCamera ( )
```

Construct a new User Camera object.

4.28.1.2 ~UserCamera()

```
Gui::UserCamera::~UserCamera ( )  [default]
```

Destroy the User Camera object.

4.28.2 Member Function Documentation

4.28.2.1 getCamera()

```
std::shared_ptr< Camera > Gui::UserCamera::getCamera ( )
```

Get the Camera object.

Returns

Camera - camera

4.28.2.2 getFovy()

```
float Gui::UserCamera::getFovy (
            void  ) const
```

Get the Fovy object.

Returns

float - fovy

4.28.2.3 getPlayerId()

```
size_t Gui::UserCamera::getPlayerId ( ) const
```

Get the Player Id object.

Returns

size_t - Player id.

4.28.2.4 getPosition()

```
Vector3 Gui::UserCamera::getPosition (
            void ) const
```

Get the Position object.

Returns

Vector3 - position

4.28.2.5 getTarget()

```
Vector3 Gui::UserCamera::getTarget (
            void ) const
```

Get the Target object.

Returns

Vector3 - target

4.28.2.6 getTilePos()

```
std::pair< std::size_t, std::size_t > Gui::UserCamera::getTilePos ( ) const
```

Get the Tile position object.

Returns

std::pair<std::size_t, std::size_t> - Position of the tile.

4.28.2.7 getType()

```
Gui::UserCamera::CameraType Gui::UserCamera::getType ( ) const
```

Get the Type object.

Returns

CameraType - Camera type.

4.28.2.8 getUp()

```
Vector3 Gui::UserCamera::getUp (
            void  ) const
```

Get the Up object.

Returns

Vector3 - up

4.28.2.9 isPlayerPov()

```
bool Gui::UserCamera::isPlayerPov ( ) const
```

Check if the camera is in player pov.

Returns

true - Camera is in player pov.

false - Camera is not in player pov.

Note

The player pov is the first person, second person and third person.

4.28.2.10 setFovy()

```
void Gui::UserCamera::setFovy (
            float fovy )
```

Set the Fovy object.

Parameters

fovy	New camera fovy.

4.28.2.11 setPlayerId()

```
void Gui::UserCamera::setPlayerId (
            size_t playerId )
```

Set the Player Id object.

Parameters

playerId	Player id to set.

4.28.2.12 setPosition()

```
void Gui::UserCamera::setPosition (
            Vector3 position )
```

Set the Position object.

Parameters

position	New camera position.

4.28.2.13 setTarget()

```
void Gui::UserCamera::setTarget (
            Vector3 target )
```

Set the Target object.

Parameters

target	New camera target.

4.28.2.14 setTilePos()

```
void Gui::UserCamera::setTilePos (
            std::pair< std::size_t, std::size_t > pos )
```

Set the Tile Pos object.

Parameters

pos	Position of the tile.

4.28.2.15 setType()

```
void Gui::UserCamera::setType (
            CameraType type )
```

Set the Type object.

Parameters

type	Type to set.

4.28.2.16 setUp()

```
void Gui::UserCamera::setUp (
            Vector3 up )
```

Set the Up object.

Parameters

up	New camera up vector.

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

- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/include/Render/UserCamera.hpp
- /home/tjerome-rocher/Desktop/Tek2/Zappy/gui/src/Render/UserCamera.cpp

Chapter 5

File Documentation

5.1 Assets.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy GUI
00004 ** File description:
00005 ** Assets
00006 */
00007
00008 #pragma once
00009
00010 #define PATH_ASSETS                 "gui/assets/"
00011
00012 #define PATH_RESOURCES              "resources/"
00013 #define PATH_PLAYER                 "player/"
00014 #define PATH_TILE                   "tile/"
00015 #define PATH_DECORATION             "decoration/"
00016 #define PATH_HUD                    "hud/"
00017
00018 #define PNG_CURSOR                  PATH_ASSETS PATH_HUD "cursor.png"
00019 #define PNG_HUD                     PATH_ASSETS PATH_HUD "hud.png"
00020 #define FONT_HUD                    PATH_ASSETS PATH_HUD "SimplyMono-Bold.ttf"
00021
00022 #define MODEL_TILE                  PATH_ASSETS PATH_TILE "tile.glb"
00023 #define MODEL_FOOD                  PATH_ASSETS PATH_RESOURCES "food.glb"
00024 #define MODEL_LINEMATE              PATH_ASSETS PATH_RESOURCES "linemate.glb"
00025 #define MODEL_MENDIANE              PATH_ASSETS PATH_RESOURCES "mendiane.glb"
00026 #define MODEL_PHIRAS                PATH_ASSETS PATH_RESOURCES "phiras.glb"
00027 #define MODEL_SIBUR                 PATH_ASSETS PATH_RESOURCES "sibur.glb"
00028 #define MODEL_THYSTAME              PATH_ASSETS PATH_RESOURCES "thystame.glb"
00029 #define MODEL_DERAUMERE             PATH_ASSETS PATH_RESOURCES "deraumere.glb"
00030 #define MODEL_PLAYER                PATH_ASSETS PATH_PLAYER "player.glb"
00031 #define MODEL_EGG                   PATH_ASSETS PATH_PLAYER "egg.glb"
00032 #define MODEL_TREE                  PATH_ASSETS PATH_DECORATION "tree.glb"
00033 #define MODEL_LANTERN               PATH_ASSETS PATH_DECORATION "lantern.glb"
00034
00035 #define SCALE_FOOD                  (Vector3){1, 0.5, 1}
00036 #define SCALE_LINEMATE              (Vector3){0.1, 0.1, 0.1}
00037 #define SCALE_MENDIANE              (Vector3){0.1, 0.1, 0.1}
00038 #define SCALE_PHIRAS                (Vector3){0.001, 0.001, 0.005}
00039 #define SCALE_SIBUR                 (Vector3){0.01, 0.01, 0.01}
00040 #define SCALE_THYSTAME              (Vector3){2, 2, 2}
00041 #define SCALE_DERAUMERE             (Vector3){0.5, 0.5, 0.5}
00042 #define SCALE_PLAYER                (Vector3){0.5, 0.5, 0.5}
00043 #define SCALE_EGG                   (Vector3){0.5, 0.5, 0.5}
00044 #define SCALE_TREE                  (Vector3){1, 1, 1}
00045 #define SCALE_LANTERN               (Vector3){1, 1, 1}
00046
00047 #define ROTATION_ANGLE_FOOD         45
00048 #define ROTATION_ANGLE_LINEMATE     270
00049 #define ROTATION_ANGLE_MENDIANE     0
00050 #define ROTATION_ANGLE_PHIRAS       270
00051 #define ROTATION_ANGLE_SIBUR        270
00052 #define ROTATION_ANGLE_THYSTAME     0
00053 #define ROTATION_ANGLE_DERAUMERE    0
00054 #define ROTATION_ANGLE_PLAYER       0
00055 #define ROTATION_ANGLE_EGG          270
00056 #define ROTATION_ANGLE_TREE         0
00057 #define ROTATION_ANGLE_LANTERN      0
00058
```

```
00059 #define ROTATION_AXIS_FOOD          (Vector3){0, 1, 0}
00060 #define ROTATION_AXIS_LINEMATE      (Vector3){1, 0, 0}
00061 #define ROTATION_AXIS_MENDIANE      (Vector3){1, 0, 0}
00062 #define ROTATION_AXIS_PHIRAS        (Vector3){1, 0, 0}
00063 #define ROTATION_AXIS_SIBUR         (Vector3){1, 0, 0}
00064 #define ROTATION_AXIS_THYSTAME      (Vector3){0, 0, 0}
00065 #define ROTATION_AXIS_DERAUMERE     (Vector3){0, 0, 0}
00066 #define ROTATION_AXIS_PLAYER        (Vector3){0, 1, 0}
00067 #define ROTATION_AXIS_EGG           (Vector3){1, 0, 0}
00068 #define ROTATION_AXIS_TREE          (Vector3){1, 0, 0}
00069 #define ROTATION_AXIS_LANTERN       (Vector3){1, 0, 0}
00070
00071 #define POS_FOOD                    (Vector3){0.5, -0.1, 1.5}
00072 #define POS_LINEMATE                (Vector3){1, -0.3, -0.5}
00073 #define POS_MENDIANE                (Vector3){2, -0.25, -0.5}
00074 #define POS_PHIRAS                  (Vector3){0.5, -0.3, -1.5}
00075 #define POS_SIBUR                   (Vector3){1.5, -0.3, -1.5}
00076 #define POS_THYSTAME                (Vector3){1, -0.2, -2}
00077 #define POS_DERAUMERE               (Vector3){2, -0.3, -2}
00078 #define POS_PLAYER                  (Vector3){0, -0.25, 0}
00079 #define POS_EGG                     (Vector3){0.5, 0, 0.5}
00080 #define POS_TREE                    (Vector3){2, -0.3, 2}
00081 #define POS_LANTERN                 (Vector3){1, -0.3, 2}
```

5.2 Colors.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy
00004 ** File description:
00005 ** Colors
00006 */
00007
00008 #pragma once
00009
00010 #define STR_BLUE "\033[0;34m"
00011 #define STR_GREEN "\033[0;32m"
00012 #define STR_RED "\033[0;31m"
00013 #define STR_YELLOW "\033[0;33m"
00014 #define STR_VIOLET "\033[0;35m"
00015 #define STR_CYAN "\033[0;36m"
00016 #define STR_RESET "\033[0m"
```

5.3 Config.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy
00004 ** File description:
00005 ** Config
00006 */
00007
00008 #pragma once
00009
00010 #define SIZE_TILE                   4.7
00011
00012 #define PLAYER_HEIGHT               2
00013
00014 #define DEFAULT_RENDER_DISTANCE     5
00015 #define MAX_RENDER_DISTANCE         15
00016 #define MIN_RENDER_DISTANCE         1
00017
00018 #define HIGH_CAMERA_INCREASE 0.1
00019 #define LOW_CAMERA_INCREASE 0.1
00020
00021 #define PLAYER_SECOND_PERSON_FOV 4.0f
00022 #define PLAYER_THIRD_PERSON_FOV 5.0f
```

5.4 Engine.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy GUI
00004 ** File description:
00005 ** Engine
```

```
00006 */
00007
00008 #pragma once
00009
00010 #include "Event/Event.hpp"
00011 #include "Render/Render.hpp"
00012 #include "Network/Network.hpp"
00013 #include "GameDatas/GameData.hpp"
00014 #include "Parsing/ServerParser.hpp"
00015 #include "GUIUpdater/GUIUpdater.hpp"
00016
00017 #include <time.h>
00018
00019 namespace Gui {
00020
00025     class Engine;
00026 };
00027
00028 class Gui::Engine {
00029
00030     public:
00031
00037         Engine(std::shared_ptr<Network> network);
00038
00043         ~Engine() = default;
00044
00049         void run(void);
00050
00051     private:
00052
00053         ServerParser                _parser;        // Parser class for server's command.
00054         std::shared_ptr<Network>    _network;       // Network class to connect to the server.
00055         std::shared_ptr<Render>     _render;        // Render class to draw the scene.
00056         Event                       _event;         // Event class to listen the user's inputs.
00057         std::shared_ptr<GameData>   _gameData;      // GameData class to store the game's data.
00058         GUIUpdater                  _guiUpdater;    // GUIUpdater class to update the GUI.
00059
00064         void listenServer(void);
00065
00073         void sendMessageUpdate(void);
00074 };
```

5.5 AError.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy GUI
00004 ** File description:
00005 ** AError
00006 */
00007
00008
00009 #pragma once
00010
00011 #include "Error/IError.hpp"
00012
00013 #include <string>
00014
00015 namespace Gui {
00016     namespace Errors {
00023         class AError;
00024     };
00025 };
00026
00027 class Gui::Errors::AError : public IError {
00028
00029     public:
00030
00034         ~AError() override = default;
00035
00040         const char *what() const noexcept override;
00041
00042     protected:
00043
00044         std::string _message;
00045 };
```

5.6 Error.hpp

```
00001 /*
```

```
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy GUI
00004 ** File description:
00005 ** Error
00006 */
00007
00008 #pragma once
00009
00010 #include "Error/AError.hpp"
00011
00012 namespace Gui {
00013     namespace Errors {
00014
00019         class Error : public AError {};
00020
00025         class NetworkException : public Error {
00026
00027             public:
00033                 NetworkException(std::string message);
00034         };
00035
00040         class ServerParserException : public Error {
00041
00042             public:
00048                 ServerParserException(std::string message);
00049         };
00050
00055         class ParseCommandLineException : public Error {
00056
00057             public:
00063                 ParseCommandLineException(std::string message);
00064         };
00065
00070         class GuiGameDataException : public Error {
00071
00072             public:
00078                 GuiGameDataException(std::string message);
00079         };
00080
00085         class GuiUpdaterException : public Error {
00086
00087             public:
00093                 GuiUpdaterException(std::string message);
00094         };
00095     };
00096 };
```

5.7 IError.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy GUI
00004 ** File description:
00005 ** IError
00006 */
00007
00008 #pragma once
00009
00010 #include <exception>
00011
00012 namespace Gui {
00013     namespace Errors {
00019         class IError;
00020     };
00021 };
00022
00023 class Gui::Errors::IError : public std::exception {
00024
00025     public:
00026
00030         virtual ~IError() = default;
00031
00039         virtual const char *what() const noexcept = 0;
00040 };
```

5.8 Event.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
```

```
00003 ** Zappy GUI
00004 ** File description:
00005 ** Event
00006 */
00007
00008 #pragma once
00009
00010 #include "Config.hpp"
00011 #include "Render/Render.hpp"
00012
00013 #include <functional>
00014 #include <unordered_map>
00015
00016 namespace Gui {
00017
00022     class Event;
00023 };
00024
00025 class Gui::Event {
00026
00027     public:
00028
00033         Event();
00034
00039         ~Event() = default;
00040
00045         void listen();
00046
00052         void setRender(std::shared_ptr<Render> render);
00053
00059         void setGameData(std::shared_ptr<GameData> gameData);
00060
00061     private:
00062
00063         std::shared_ptr<Render>         _render;    // Render class to draw scene.
00064         std::shared_ptr<GameData>       _gameData;  // GameData class to contain scene.
00065
00070         std::unordered_map<KeyboardKey, std::function<void()» _eventsKeyDown =
00071         {
00072             {KEY_SPACE, [this](){moveUpCamera();}},
00073             {KEY_LEFT_SHIFT, [this](){moveDownCamera();}},
00074         };
00075
00080         std::unordered_map<KeyboardKey, std::function<void()» _eventsKeyPressed =
00081         {
00082             {KEY_THREE, [this](){switchDisplayDebug();}},
00083             {KEY_F3, [this](){switchDisplayDebug();}},
00084             {KEY_SPACE, [this](){setFreeCam();}},
00085             {KEY_R, [this](){switchTileHudToGame();}},
00086             {KEY_J, [this](){increaseRenderDistance();}},
00087             {KEY_K, [this](){decreaseRenderDistance();}},
00088             {KEY_F5, [this](){changeActualPlayerPov();}},
00089             {KEY_APOSTROPHE, [this](){changeActualPlayerPov();}},
00090             {KEY_KP_ADD, [this](){increaseTimeUnit();}},
00091             {KEY_KP_SUBTRACT, [this](){decreaseTimeUnit();}},
00092         };
00093
00098         std::unordered_map<MouseButton, std::function<void()» _eventsMousePressed =
00099         {
00100             {MOUSE_BUTTON_LEFT, [this](){handleLeftClick();}},
00101             {MOUSE_BUTTON_RIGHT, [this](){handleRightClick();}}
00102         };
00103
00108         void moveUpCamera();
00109
00114         void moveDownCamera();
00115
00120         void switchDisplayDebug();
00121
00125         void setFreeCam();
00126
00131         void handleLeftClick();
00132
00136         void handleRightClick();
00137
00142         void selectPlayer();
00143
00147         void selectTile();
00148
00154         void changePlayer(bool turn);
00155
00162         void changePlayerPOV(size_t playerId);
00163
00169         void setPlayerPov(size_t playerId);
00170
00175         void changeActualPlayerPov();
00176
```

```
00182         void changePOVToFirstPerson(size_t id);
00183
00189         void changePOVToSecondPerson(size_t id);
00190
00196         void changePOVToThirdPerson(size_t id);
00197
00202         void switchTileHudToGame();
00203
00208         void increaseRenderDistance();
00209
00214         void decreaseRenderDistance();
00215
00220         void increaseTimeUnit();
00221
00226         void decreaseTimeUnit();
00227 };
```

5.9 Egg.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy
00004 ** File description:
00005 ** Egg
00006 */
00007
00008 #pragma once
00009
00010 #include "Error/Error.hpp"
00011
00012 #include <string>
00013
00014 namespace Gui {
00015
00019     class Egg;
00020 };
00021
00022 class Gui::Egg {
00023
00024     public:
00025
00026         enum EggState {
00027             IDLE,
00028             DEAD,
00029             BORN,
00030             HATCHING,
00031         };
00032
00043         Egg(size_t id, const std::string &team, std::pair<std::size_t, std::size_t> position);
00044
00050         ~Egg();
00051
00057         std::size_t getId() const;
00058
00064         std::string getTeam() const;
00065
00071         std::pair<std::size_t, std::size_t> getPosition() const;
00072
00078         void setId(std::size_t id);
00079
00085         void setTeam(const std::string &team);
00086
00092         void setPosition(std::pair<std::size_t, std::size_t> position);
00093
00099         void setState(EggState state);
00100
00106         EggState getState() const;
00107
00108     private:
00109
00110         std::size_t                             _id;        // Id of the egg.
00111         std::string                             _team;      // Team name of the egg.
00112         std::pair<std::size_t, std::size_t>     _position;  // Position of the egg.
00113         EggState                                _state;     // State of the egg.
00114 };
```

5.10 GameData.hpp

```
00001 /*
```

```
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy
00004 ** File description:
00005 ** GameData
00006 */
00007
00008 #pragma once
00009
00010 #include "Types.hpp"
00011 #include "Error/Error.hpp"
00012 #include "GameDatas/Team.hpp"
00013 #include "GameDatas/Tile.hpp"
00014
00015 #define NO_TICK int(-1)
00016
00017 namespace Gui {
00018
00023     class GameData;
00024 };
00025
00026 class Gui::GameData {
00027
00028     public:
00029
00030         enum TimeUnitState {
00031             INCREASE,
00032             DECREASE,
00033             NONE
00034         };
00035
00040         GameData();
00041
00046         ~GameData() = default;
00047
00053         std::vector<Gui::Team> &getTeams();
00054
00061         Gui::Team &getTeam(const std::string &name);
00062
00068         void addTeam(const Gui::Team &team);
00069
00077         void addTeam(const std::string &name, const std::string &playerModelPath, const std::string
      &eggModelPath);
00078
00085         void addPlayerToTeam(const std::string &teamName, const Gui::Player &player);
00086
00092         Gui::Player &getPlayer(size_t id);
00093
00099         Map<Gui::Tile> &getMap();
00100
00106         void setMap(const Map<Gui::Tile> &map);
00107
00115         void setMapSize(size_t x, size_t y);
00116
00122         std::pair<size_t, size_t> getMapSize() const;
00123
00131         Gui::Tile &getTile(size_t x, size_t y);
00132
00140         void setTile(const Gui::Tile &tile);
00141
00146         void restartLastTick(void);
00147
00153         void setServerTick(std::size_t tick);
00154
00160         clock_t getLastTick() const;
00161
00167         std::size_t getServerTick() const;
00168
00174         void setIsEndGame(bool isEndGame);
00175
00182         bool getIsEndGame() const;
00183
00189         void setLastError(const std::string &error);
00190
00196         std::string getLastError() const;
00197
00204         Team &getTeamById(std::size_t id);
00205
00211         TimeUnitState getTimeUnitFromServer() const;
00212
00218         void setTimeUnitFromServer(TimeUnitState timeUnitFromServer);
00219
00220     private:
00221
00222         std::vector<Gui::Team>      _teams;         // Teams of the game.
00223         Map<Gui::Tile>              _map;           // Map of the game.
00224         std::size_t                 _serverTick;    // Tick value of the server.
00225         clock_t                     _lastTick;      // Last tick of the GameData (based on the server
```

```
    tick).
00226         bool                        _isEndGame;     // Is true if the game is finished.
00227         std::string                 _lastError;     // Last error message.
00228         TimeUnitState                       _timeUnitFromServer; // True if the time unit has
    changed.
00229 };
```

5.11 Inventory.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy GUI
00004 ** File description:
00005 ** Inventory
00006 */
00007
00008 #pragma once
00009
00010 #include <iostream>
00011
00012 #define RessourcesNumber 6
00013
00014 namespace Gui {
00015
00020     class Inventory;
00021 };
00022
00023 class Gui::Inventory {
00024
00025     public:
00026
00027         using Ressources = size_t [RessourcesNumber];
00028
00033         Inventory();
00034
00046         Inventory(std::size_t food, std::size_t linemate, std::size_t deraumere, std::size_t sibur,
    std::size_t mendiane, std::size_t phiras, std::size_t thystame);
00047
00052         ~Inventory() = default;
00053
00059         void setFood(std::size_t food);
00060
00066         void setLinemate(std::size_t linemate);
00067
00073         void setDeraumere(std::size_t deraumere);
00074
00080         void setSibur(std::size_t sibur);
00081
00087         void setMendiane(std::size_t mendiane);
00088
00094         void setPhiras(std::size_t phiras);
00095
00101         void setThystame(std::size_t thytsame);
00102
00108         void setRessources(Ressources ressources);
00109
00115         std::size_t getFood(void);
00116
00122         std::size_t getLinemate(void);
00123
00129         std::size_t getDeraumere(void);
00130
00136         std::size_t getSibur(void);
00137
00143         std::size_t getMendiane(void);
00144
00150         std::size_t getPhiras(void);
00151
00157         std::size_t getThystame(void);
00158
00164         Ressources &getRessources(void);
00165
00166     private:
00167
00168         std::size_t     _food;    // Food.
00169         Ressources      _ressources;    // Ressources.
00170 };
```

5.12 Player.hpp

```
00001 /*
```

```
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy GUI
00004 ** File description:
00005 ** Player
00006 */
00007
00008 #pragma once
00009
00010 #include "raylib.h"
00011 #include "GameDatas/Inventory.hpp"
00012
00013 namespace Gui {
00014
00019     class Player;
00020 };
00021
00022 class Gui::Player {
00023
00024     public:
00025
00026         enum PlayerState {
00027             IDLE,
00028             BORN,
00029             BROADCAST,
00030             EJECT,
00031             BEING_EJECTED,
00032             WALK,
00033             INCANTATION,
00034             FINISHED_INCANTATION,
00035             LAY_EGG,
00036             DROP,
00037             COLLECT,
00038             DEAD,
00039         };
00040
00048         Player(std::size_t id, const std::string &team, std::pair<std::size_t, std::size_t> position,
      std::size_t orientation, std::size_t level = 1);
00049
00054         ~Player() = default;
00055
00061         void setPosition(std::pair<std::size_t, std::size_t> position);
00062
00068         void setId(std::size_t id);
00069
00075         void setLevel(std::size_t level);
00076
00082         void setOrientation(std::size_t orientation);
00083
00089         void setTeam(const std::string &team);
00090
00096         std::pair<std::size_t, std::size_t> getPosition(void) const;
00097
00103         std::size_t getId(void) const;
00104
00110         std::size_t getLevel(void) const;
00111
00117         std::size_t getOrientation(void) const;
00118
00124         std::string getTeam(void) const;
00125
00131         void setState(PlayerState state);
00132
00138         PlayerState getState(void) const;
00139
00145         void setBroadcast(const std::string &broadcast);
00146
00152         std::string getBroadcast() const;
00153
00159         float getRotationFromOrientation() const;
00160
00166         Vector3 getCenterPosition();
00167
00172         Inventory                               inventory;
00173
00174     private:
00175
00176         std::size_t                             _id;            // Id of the player.
00177         std::string                             _team;          // Team name.
00178         std::pair<std::size_t, std::size_t>     _position;      // Position x y.
00179         std::size_t                             _orientation;   // Orientation of the player.
00180         std::size_t                             _level;         // Level between 1 - 8.
00181         PlayerState                             _state;         // Player state.
00182         std::string                             _broadcast;     // Broadcast message.
00183 };
```

5.13 Team.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy
00004 ** File description:
00005 ** Team
00006 */
00007
00008 #pragma once
00009
00010 #include "raylib.h"
00011 #include "Types.hpp"
00012 #include "GameDatas/Egg.hpp"
00013 #include "GameDatas/Tile.hpp"
00014 #include "GameDatas/Player.hpp"
00015
00016 #include <vector>
00017 #include <memory>
00018
00019 namespace Gui {
00020
00025     class Team;
00026 };
00027
00028 class Gui::Team {
00029
00030     public:
00031
00039         Team(const std::string &name, const std::string &playerModelPath, const std::string
      &eggModelPath);
00040
00045         ~Team();
00046
00052         const std::string &getName() const;
00053
00059         std::vector<Gui::Player> &getPlayers();
00060
00066         std::vector<Gui::Egg> &getEggs();
00067
00073         void setName(const std::string &name);
00074
00080         void addPlayer(const Gui::Player &player);
00081
00087         void addEgg(const Gui::Egg &egg);
00088
00096         bool removePlayer(std::size_t id);
00097
00105         bool removeEgg(std::size_t id);
00106
00113         std::shared_ptr<Gui::Player> getPlayer(std::size_t id);
00114
00120         Model getPlayerModel() const;
00121
00127         void setPlayerModelPath(const std::string &playerModelPath);
00128
00135         std::shared_ptr<Gui::Egg> getEgg(std::size_t id);
00136
00142         Model getEggModel() const;
00143
00149         void setEggModelPath(const std::string &eggModelPath);
00150
00158         std::vector<BoundingBox> getPlayerBoundingBoxes(std::pair<size_t, size_t> pos, size_t
      orientation, Vector3 center);
00159
00166         Vector3 getPlayerPositionIn3DSpace(size_t id, Map<Tile> map);
00167
00175         std::vector<RayCollision> getPlayerModelHitbox(size_t id, Camera camera);
00176
00184         bool isPlayerHit(size_t id, Camera camera);
00185
00186     private:
00187
00188         std::string                 _name;      // Name of the team.
00189         std::vector<Gui::Player>    _players;   // Players of the team.
00190         Model                       _playerModel;     // Model player asset of the team.
00191         std::vector<Gui::Egg>       _eggs;      // Eggs of the team.
00192         Model                       _eggModel;  // Eggs Model of the team.
00193
00203         BoundingBox rotateBoundingBoxByOrientation(BoundingBox bbox, size_t orientation,
      std::pair<size_t, size_t> pos, Vector3 center);
00204 };
```

5.14 Tile.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy GUI
00004 ** File description:
00005 ** Tile
00006 */
00007
00008 #pragma once
00009
00010 #include "raylib.h"
00011 #include "GameDatas/Inventory.hpp"
00012
00013 #include <vector>
00014
00015 namespace Gui {
00016
00021     class Tile;
00022 };
00023
00024 class Gui::Tile {
00025
00026     public:
00027
00033         Tile(std::pair<std::size_t, std::size_t> position);
00034
00041         Tile(std::pair<std::size_t, std::size_t> position, Inventory inventory);
00042
00047         ~Tile() = default;
00048
00054         void setPosition(std::pair<std::size_t, std::size_t> position);
00055
00061         std::pair<std::size_t, std::size_t> getPosition() const;
00062
00068         Vector3 getPositionIn3DSpace();
00069
00076         std::vector<BoundingBox> getTileBoundingBoxes(Tile tile, Model tileModel);
00077
00085         std::vector<RayCollision> getTileModelHitbox(Tile tile, Camera camera, Model tileModel);
00086
00095         bool isTileHit(Camera camera, Model _tileModel);
00096
00101         Inventory       inventory;
00102
00103     private:
00104
00105         std::pair<std::size_t, std::size_t>     _position;              // Position x y.
00106         Vector3                                 _positionIn3DSpace;     // Position in 3D space.
00107 };
```

5.15 GUIUpdater.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy
00004 ** File description:
00005 ** GUIUpdater
00006 */
00007
00008 #pragma once
00009
00010 #include "Network/Network.hpp"
00011 #include "GameDatas/GameData.hpp"
00012
00013 #include <string>
00014 #include <functional>
00015 #include <unordered_map>
00016
00017 namespace Gui {
00018
00024     class GUIUpdater;
00025 }
00026
00027 class Gui::GUIUpdater {
00028
00029     public:
00030
00037         GUIUpdater(std::shared_ptr<GameData> gameData, std::shared_ptr<Network> network);
00038
00042         ~GUIUpdater() = default;
00043
00050         void update(const std::string &command, const std::vector<std::string> &data);
```

```
00051
00052     private:
00053
00054         std::shared_ptr<GameData> _gameData; // The GUI GameData to update.
00055         std::shared_ptr<Network> _network; // The network to send commands to the server.
00056
00057         std::unordered_map<std::string, std::function<void(std::vector<std::string>)» _updateMap =
00058         {
00059             {"msz", [this](std::vector<std::string> data){updateMapSize(data);}},
00060             {"bct", [this](std::vector<std::string> data){updateMapContent(data);}},
00061             {"tna", [this](std::vector<std::string> data){updateTeamNames(data);}},
00062             {"pnw", [this](std::vector<std::string> data){updateTeamMember(data);}},
00063             {"ppo", [this](std::vector<std::string> data){updatePlayerPosition(data);}},
00064             {"plv", [this](std::vector<std::string> data){updatePlayerLevel(data);}},
00065             {"pin", [this](std::vector<std::string> data){updatePlayerInventory(data);}},
00066             {"pex", [this](std::vector<std::string> data){updatePlayerExpulsion(data);}},
00067             {"pbc", [this](std::vector<std::string> data){updatePlayerBroadcast(data);}},
00068             {"pic", [this](std::vector<std::string> data){updatePlayerStartIncantation(data);}},
00069             {"pie", [this](std::vector<std::string> data){updatePlayerEndIncantation(data);}},
00070             {"pfk", [this](std::vector<std::string> data){updatePlayerEggLaying(data);}},
00071             {"pdr", [this](std::vector<std::string> data){updatePlayerRessourceDropping(data);}},
00072             {"pgt", [this](std::vector<std::string> data){updatePlayerRessourceCollecting(data);}},
00073             {"pdi", [this](std::vector<std::string> data){updatePlayerDeath(data);}},
00074             {"enw", [this](std::vector<std::string> data){updateEggLaidByPlayer(data);}},
00075             {"ebo", [this](std::vector<std::string> data){updatePlayerBorn(data);}},
00076             {"edi", [this](std::vector<std::string> data){updateEggDeath(data);}},
00077             {"sgt", [this](std::vector<std::string> data){updateTimeUnitRequest(data);}},
00078             {"sst", [this](std::vector<std::string> data){updateTimeUnitModification(data);}},
00079             {"seg", [this](std::vector<std::string> data){updateEndOfGame(data);}},
00080             {"smg", [this](std::vector<std::string> data){updateMessageFromServer(data);}},
00081             {"suc", [this](std::vector<std::string> data){updateUnknownMessage(data);}},
00082             {"sbp", [this](std::vector<std::string> data){updateCommandParameter(data);}},
00083         }; // The map of commands to update the GUI GameData.
00084
00090         void updateMapSize(const std::vector<std::string> &data);
00091
00097         void updateMapContent(const std::vector<std::string> &data);
00098
00104         void updateTeamNames(const std::vector<std::string> &data);
00105
00111         void updateTeamMember(const std::vector<std::string> &data);
00112
00118         void updatePlayerPosition(const std::vector<std::string> &data);
00119
00125         void updatePlayerLevel(const std::vector<std::string> &data);
00126
00132         void updatePlayerInventory(const std::vector<std::string> &data);
00133
00139         void updatePlayerExpulsion(const std::vector<std::string> &data);
00140
00146         void updatePlayerBroadcast(const std::vector<std::string> &data);
00147
00153         void updatePlayerStartIncantation(const std::vector<std::string> &data);
00154
00160         void updatePlayerEndIncantation(const std::vector<std::string> &data);
00161
00167         void updatePlayerEggLaying(const std::vector<std::string> &data);
00168
00174         void updatePlayerRessourceDropping(const std::vector<std::string> &data);
00175
00181         void updatePlayerRessourceCollecting(const std::vector<std::string> &data);
00182
00188         void updatePlayerDeath(const std::vector<std::string> &data);
00189
00195         void updateEggLaidByPlayer(const std::vector<std::string> &data);
00196
00202         void updatePlayerBorn(const std::vector<std::string> &data);
00203
00209         void updateEggDeath(const std::vector<std::string> &data);
00210
00216         void updateTimeUnitRequest(const std::vector<std::string> &data);
00217
00223         void updateTimeUnitModification(const std::vector<std::string> &data);
00224
00230         void updateEndOfGame(const std::vector<std::string> &data);
00231
00237         void updateMessageFromServer(const std::vector<std::string> &data);
00238
00244         void updateUnknownMessage(const std::vector<std::string> &data);
00245
00251         void updateCommandParameter(const std::vector<std::string> &data);
00252 };
```

5.16 AHud.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy GUI
00004 ** File description:
00005 ** AHud
00006 */
00007
00008 #pragma once
00009
00010 #include "Hud/IHud.hpp"
00011 #include "GameDatas/GameData.hpp"
00012
00013 namespace Gui {
00014
00019     class AHud;
00020 };
00021
00022 class Gui::AHud : public Gui::IHud {
00023
00024     public:
00025
00030         ~AHud() = default;
00031
00036         virtual void display() = 0;
00037
00043         void setPlayer(std::shared_ptr<Player> player);
00044
00050         void setTile(std::shared_ptr<Tile> tile);
00051
00057         TypeScene getType() const;
00058
00059     protected:
00060
00061         TypeScene                       _typeScene;     // Type of the scene.
00062         std::shared_ptr<GameData>       _gameData;      // GameData class.
00063         std::shared_ptr<Player>         _player;        // Player to display hud.
00064         std::shared_ptr<Tile>           _tile;          // Tile to display hud.
00065 };
```

5.17 HudGame.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy GUI
00004 ** File description:
00005 ** HudGame
00006 */
00007
00008 #pragma once
00009
00010 #include "Hud/AHud.hpp"
00011
00012 #define HUD_GAME_POS Vector2{0, 340}
00013 #define HUD_GAME_TEXT_POS Vector2{25, 420}
00014 #define HUD_GAME_TEXT_TITLE_POS Vector2{50, 380}
00015 #define HUD_GAME_TEXT_MARGING 30
00016
00017 namespace Gui {
00018
00023     class HudGame;
00024 };
00025
00026 class Gui::HudGame : public Gui::AHud {
00027
00028     public:
00029
00035         HudGame(std::shared_ptr<GameData> gameData);
00036
00041         ~HudGame() = default;
00042
00047         void display();
00048
00049     private:
00050
00051         Texture2D   _texture;       // Texture for Hud Background.
00052         Font        _font;          // Font for Hud's texts.
00053 };
```

5.18 HudPlayer.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy GUI
00004 ** File description:
00005 ** HudPlayer
00006 */
00007
00008 #pragma once
00009
00010 #include "Hud/AHud.hpp"
00011
00012 #define HUD_PLAYER_POS Vector2{0, 340}
00013 #define HUD_PLAYER_TEXT_POS Vector2{25, 420}
00014 #define HUD_PLAYER_TEXT_TITLE_POS Vector2{50, 380}
00015 #define HUD_PLAYER_TEXT_MARGING 30
00016
00017 namespace Gui {
00018
00023     class HudPlayer;
00024 };
00025
00026 class Gui::HudPlayer : public Gui::AHud {
00027
00028     public:
00029
00035         HudPlayer(std::shared_ptr<GameData> gameData);
00036
00041         ~HudPlayer() = default;
00042
00047         void display();
00048
00049     private:
00050
00051         Texture2D   _texture;       // Texture for Hud Background.
00052         Font        _font;          // Font for Hud's texts.
00053 };
```

5.19 HudTile.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy GUI
00004 ** File description:
00005 ** HudTile
00006 */
00007
00008 #pragma once
00009
00010 #include "Hud/AHud.hpp"
00011
00012 #define HUD_TILE_POS Vector2{0, 340}
00013 #define HUD_TILE_TEXT_POS Vector2{25, 420}
00014 #define HUD_TILE_TEXT_TITLE_POS Vector2{50, 380}
00015 #define HUD_TILE_TEXT_MARGING 30
00016
00017 namespace Gui {
00018
00023     class HudTile;
00024 };
00025
00026 class Gui::HudTile : public Gui::AHud {
00027
00028     public:
00029
00035         HudTile(std::shared_ptr<GameData> gameData);
00036
00041         ~HudTile() = default;
00042
00047         void display();
00048
00049     private:
00050
00051         Texture2D   _texture;       // Texture for Hud Background.
00052         Font        _font;          // Font for Hud's texts.
00053 };
```

5.20 IHud.hpp

```
00001 /*
```

```
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy GUI
00004 ** File description:
00005 ** IHud
00006 */
00007
00008 #pragma once
00009
00010 #include "GameDatas/Player.hpp"
00011 #include "GameDatas/Tile.hpp"
00012
00013 #include <memory>
00014
00015 namespace Gui {
00016
00021     class IHud;
00022 };
00023
00024 class Gui::IHud {
00025
00026     public:
00027
00032         enum TypeScene {
00033             GAME,
00034             POV_PLAYER,
00035             END_GAME,
00036             TILE
00037         };
00038
00043         virtual ~IHud() = default;
00044
00049         virtual void display() = 0;
00050
00056         virtual void setPlayer(std::shared_ptr<Player> player) = 0;
00057
00063         virtual void setTile(std::shared_ptr<Tile> tile) = 0;
00064
00070         virtual TypeScene getType() const = 0;
00071 };
```

5.21 Network.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy GUI
00004 ** File description:
00005 ** Network
00006 */
00007
00008 #pragma once
00009
00010 #include "Error/Error.hpp"
00011
00012 #include <string>
00013 #include <arpa/inet.h>
00014 #include <sys/socket.h>
00015 #include <netinet/in.h>
00016
00017 #define MAX_PORT 65535
00018 #define MIN_PORT 1
00019
00020 namespace Gui {
00021
00026     class Network;
00027 };
00028
00029 class Gui::Network {
00030
00031     public:
00032
00039         Network(int port, const std::string& hostName);
00040
00045         ~Network() = default;
00046
00052         void setPort(int port);
00053
00059         void setHostName(const std::string& hostName);
00060
00066         int getPort() const;
00067
00073         std::string getHostName() const;
00074
00079         void connectToServer();
```

```
00080
00086         const std::string listenServer();
00087
00093         void sendMessageServer(const std::string& message);
00094
00095     private:
00096
00101         void selectServer();
00102
00108         const std::string readInfoServer();
00109
00110         int             _port;          // server port
00111         std::string     _hostName;      // server hostname
00112         int             _serverFd;      // server file descriptor
00113         fd_set          _writeFd;       // file descriptor for write access
00114         fd_set          _readFd;        // file descriptor for read access
00115         bool            _isConnected;   // is true if the gui is connected to the server
00116 };
```

5.22 ParseCommandLine.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy GUI
00004 ** File description:
00005 ** ParseCommandLine
00006 */
00007
00008 #pragma once
00009
00010 #include <string>
00011
00012 #define GUI_USAGE "USAGE:\t./zappy_gui -p port -h machine"
00013
00014 namespace Gui {
00015
00020     class ParseCommandLine;
00021 };
00022
00023 class Gui::ParseCommandLine {
00024
00025     public:
00026
00033         ParseCommandLine(int argc, char **argv);
00034
00039         ~ParseCommandLine() = default;
00040
00047         void parseFlags(int argc, char **argv);
00048
00054         int getPort(void);
00055
00061         std::string getHostName(void);
00062
00063     private:
00064
00065         int             _port;      // port server
00066         std::string     _hostName;  // host name server
00067 };
```

5.23 ServerParser.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy GUI
00004 ** File description:
00005 ** Parse server command
00006 */
00007
00008 #pragma once
00009
00010 #include "Error/Error.hpp"
00011
00012 #include <string>
00013 #include <functional>
00014 #include <unordered_map>
00015
00016 namespace Gui {
00017
00022     class ServerParser;
```

```
00023 };
00024
00025 class Gui::ServerParser {
00026
00027     public:
00028
00033         ServerParser() = default;
00034
00039         ~ServerParser() = default;
00040
00047         std::vector<std::string> parse(const std::string& command);
00048
00053         enum ParseType {
00054             INT,
00055             STRING,
00056             MESSAGE,
00057             LIST_INT
00058         };
00059
00060     private:
00061
00066         std::unordered_map<std::string, std::vector<ParseType» _typesCommand =
00067         {
00068             {"msz", std::vector<ParseType>{INT, INT}},
00069             {"bct", std::vector<ParseType>{INT, INT, INT, INT, INT, INT, INT, INT, INT}},
00070             {"tna", std::vector<ParseType>{STRING}},
00071             {"pnw", std::vector<ParseType>{INT, INT, INT, INT, INT, STRING}},
00072             {"ppo", std::vector<ParseType>{INT, INT, INT, INT}},
00073             {"plv", std::vector<ParseType>{INT, INT}},
00074             {"pin", std::vector<ParseType>{INT, INT, INT, INT, INT, INT, INT, INT, INT, INT}},
00075             {"pex", std::vector<ParseType>{INT}},
00076             {"pbc", std::vector<ParseType>{INT, MESSAGE}},
00077             {"pic", std::vector<ParseType>{INT, INT, INT, LIST_INT}},
00078             {"pie", std::vector<ParseType>{INT, INT, INT}},
00079             {"pfk", std::vector<ParseType>{INT}},
00080             {"pdr", std::vector<ParseType>{INT, INT}},
00081             {"pgt", std::vector<ParseType>{INT, INT}},
00082             {"pdi", std::vector<ParseType>{INT}},
00083             {"enw", std::vector<ParseType>{INT, INT, INT, INT}},
00084             {"ebo", std::vector<ParseType>{INT}},
00085             {"edi", std::vector<ParseType>{INT}},
00086             {"sgt", std::vector<ParseType>{INT}},
00087             {"sst", std::vector<ParseType>{INT}},
00088             {"seg", std::vector<ParseType>{STRING}},
00089             {"smg", std::vector<ParseType>{MESSAGE}},
00090             {"suc", std::vector<ParseType>{}},
00091             {"sbp", std::vector<ParseType>{}}
00092         };
00093
00101         std::vector<std::string> parseCommand(const std::string& command, std::vector<ParseType>
      types);
00102
00110         std::vector<std::string> parseInt(std::istringstream& stream, std::vector<std::string>
      arguments);
00111
00119         std::vector<std::string> parseString(std::istringstream& stream, std::vector<std::string>
      arguments);
00120
00129         std::vector<std::string> parseMessage(std::istringstream& stream, std::vector<std::string>
      arguments, std::string commandName);
00130
00139         std::vector<std::string> parseListInt(std::istringstream& stream, std::vector<std::string>
      arguments, std::string commandName);
00140 };
```

5.24 Decoration.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy
00004 ** File description:
00005 ** Decoration
00006 */
00007
00008 #pragma once
00009
00010 #include "raylib.h"
00011 #include "Types.hpp"
00012 #include "Assets.hpp"
00013
00014 #include <vector>
00015 #include <iostream>
00016
```

```
00017 namespace Gui {
00018
00024     class Decoration;
00025 }
00026
00027 class Gui::Decoration {
00028
00029     public:
00030
00035         Decoration();
00036
00041         ~Decoration() = default;
00042
00050         void display(std::pair<std::size_t, std::size_t> mapSize, size_t renderDistance,
      std::pair<std::size_t, std::size_t> camPos);
00051
00060         Map<bool> getGenerationItem(std::size_t ratio);
00061
00062     private:
00063
00064         Model                                   _treeModel;         // Tree model asset.
00065         Map<bool>                               _mapTree;           // Map to display trees.
00066         Model                                   _lanternModel;      // Lantern model asset.
00067         Map<bool>                               _mapLantern;        // Map to display lanterns.
00068
00069         std::pair<std::size_t, std::size_t>     _mapSize;           // Size of the map.
00070
00078         void displayTree(size_t i, size_t j, Vector3 posTile);
00079
00087         void displayLantern(size_t i, size_t j, Vector3 posTile);
00088 };
```

5.25 Render.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy GUI
00004 ** File description:
00005 ** Render
00006 */
00007
00008 #pragma once
00009
00010 #define WINDOW_WIDTH 1920
00011 #define WINDOW_HEIGHT 1080
00012 #define WINDOW_TITLE "Zappy GUI"
00013
00014 #include "raylib.h"
00015 #include "Hud/HudGame.hpp"
00016 #include "Hud/HudTile.hpp"
00017 #include "Hud/HudPlayer.hpp"
00018 #include "Render/Decoration.hpp"
00019 #include "Render/UserCamera.hpp"
00020 #include "GameDatas/GameData.hpp"
00021
00022 #include <functional>
00023 #include <unordered_map>
00024
00025 namespace Gui {
00026
00031     class Render;
00032 };
00033
00034 class Gui::Render {
00035
00036     public:
00037
00042         Render(std::shared_ptr<GameData> gameData);
00043
00048         ~Render();
00049
00056         bool isOpen();
00057
00062         void draw();
00063
00069         std::shared_ptr<Camera> getCamera();
00070
00076         void setIsDebug(bool isDebug);
00077
00084         bool getIsDebug(void);
00085
00091         void setCameraType(Gui::UserCamera::CameraType type);
00092
```

```
00098         Gui::UserCamera::CameraType getCameraType() const;
00099
00105         void setCameraPlayerPov(std::size_t id);
00106
00112         std::size_t getCameraPlayerPov() const;
00113
00119         void setCameraTile(std::pair<std::size_t, std::size_t> pos);
00120
00126         std::pair<std::size_t, std::size_t> getCameraTile() const;
00127
00132         Model getTileModel() const;
00133
00139         void setRenderDistance(size_t renderDistance);
00140
00145         size_t getRenderDistance() const;
00146
00154         bool isCameraInPlayerPov() const;
00155
00161         size_t getTimeUnit() const;
00162
00168         void setTimeUnit(size_t timeUnit);
00169
00170     private:
00171
00172         UserCamera                                  _camera;            // Camera of the scene.
00173         bool                                        _isDebug;           // Display or not the debug
      informations.
00174         std::shared_ptr<GameData>                   _gameData;          // GameData class to store the
      game's data.
00175         std::shared_ptr<Decoration>                 _decoration;        // Decoration to display;
00176         std::vector<std::shared_ptr<Gui::IHud»     _hudList;           // List of huds.
00177         size_t                                      _renderDistance;    // Distance to render from the
      3d position of the camera.
00178
00179         Model                                       _tileModel;         // Model to display tiles.
00180         Model                                       _foodModel;         // Model to display foods.
00181         Model                                       _linemateModel;     // Model to display linemates.
00182         Model                                       _mendianeModel;     // Model to display mendianes.
00183         Model                                       _phirasModel;       // Model to display phiras.
00184         Model                                       _siburModel;        // Model to display siburs.
00185         Model                                       _thystameModel;     // Model to display thystames.
00186         Model                                       _deraumereModel;    // Model to display
      deraumeres.
00187         Texture2D                                   _cursorTexture;     // Cursor texture.
00188
00193         void LoadModels(void);
00194
00199         void displayHUD(void);
00200
00205         void displayDebug(void);
00206
00211         void displayPlayers(void);
00212
00217         void displayMap(void);
00218
00223         void displayTile(Tile tile);
00224
00230         void displayEggs(Tile tile) const;
00231
00237         void displayFood(Tile tile) const;
00238
00244         void displayResources(Tile tile) const;
00245
00251         void displayLinemate(Tile tile) const;
00252
00258         void displayMendiane(Tile tile) const;
00259
00265         void displayPhiras(Tile tile) const;
00266
00272         void displaySibur(Tile tile) const;
00273
00279         void displayThystame(Tile tile) const;
00280
00286         void displayDeraumere(Tile tile) const;
00287
00292         void displayCursor();
00293
00299         std::pair<std::size_t, std::size_t> getCameraTile();
00300 };
```

5.26 UserCamera.hpp

```
00001 /*
```

```
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy GUI
00004 ** File description:
00005 ** Camera
00006 */
00007
00008 #pragma once
00009
00010 #include "raylib.h"
00011
00012 #include <memory>
00013
00014 namespace Gui {
00015
00020     class UserCamera;
00021 };
00022
00023 class Gui::UserCamera {
00024
00025     public:
00026
00027         enum CameraType {
00028             FREE,
00029             FIRST_PERSON,
00030             SECOND_PERSON,
00031             THIRD_PERSON,
00032             FREE_TILE
00033         };
00034
00039         UserCamera();
00040
00045         ~UserCamera() = default;
00046
00052         void setPosition(Vector3 position);
00053
00059         void setTarget(Vector3 target);
00060
00066         void setUp(Vector3 up);
00067
00073         void setFovy(float fovy);
00074
00080         Vector3 getPosition(void) const;
00081
00087         Vector3 getTarget(void) const;
00088
00094         Vector3 getUp(void) const;
00095
00101         float getFovy(void) const;
00102
00108         std::shared_ptr<Camera> getCamera();
00109
00115         void setType(CameraType type);
00116
00122         CameraType getType() const;
00123
00129         void setPlayerId(size_t playerId);
00130
00136         size_t getPlayerId() const;
00137
00143         void setTilePos(std::pair<std::size_t, std::size_t> pos);
00144
00150         std::pair<std::size_t, std::size_t> getTilePos() const;
00151
00159         bool isPlayerPov() const;
00160
00161     private:
00162
00163         std::shared_ptr<Camera>                 _camera;        // Camera raylib instance.
00164         CameraType                              _type;          // Type of camera.
00165         size_t                                  _playerId;      // Player id.
00166         std::pair<std::size_t, std::size_t>     _tilePos;       // Tile position.
00167 };
```

5.27 Types.hpp

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy
00004 ** File description:
00005 ** Types
00006 */
00007
00008 #pragma once
```

```
00009
00010 #include <vector>
00011
00017 template<typename T>
00018 using Map = std::vector<std::vector<T»;
```

Index