crazyape

Generated by Doxygen 1.9.4

Chapter 1

CrazyApe

array: 0: -empty 1: -player(ape) mooves around the field according WASD 2: -red_coconut ape loose 2 lifes, moves from left to right, flies further if they touch 3: -brown_coconut ape loose 1 lifes, moves from left to rigth, flies further if they touch 4: -banana collect banana, banana removes if they touch 5: -heart increases live of ape, heart removes if they touch 6: -Tree only blocks the field 7: -Tiger moves random around the field, ape dies if the touch 8: -Scorpion moves random around the field, ape loses 1 life if the touch 9: -forest

rules: winnig: if 3(all) bananas are collected and reahes sporning point

death: -lives empty(0) -touch the tiger

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

- Collision::BitmaskManager . . . ??
- Gamecontroller . . . ??
- Gui . . . ??
- Item . . . ??
  - Animal . . . ??
    - Scorpion . . . ??
    - Tiger . . . ??
  - Banana . . . ??
  - Coconut . . . ??
  - Heart . . . ??
  - Player . . . ??
  - Tree . . . ??
- Menu . . . ??

Chapter 3

Class Index

3.1 Class List

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

Animal . . . . . ??
Banana . . . . . ??
Collision::BitmaskManager . . . . . ??
Coconut . . . . . ??
Gamecontroller . . . . . ??
Gui . . . . . ??
Heart . . . . . ??
Item . . . . . ??
Menu . . . . . ??
Player . . . . . ??
Scorpion . . . . . ??
Tiger . . . . . ??
Tree . . . . . ??

Chapter 4

File Index

4.1 File List

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

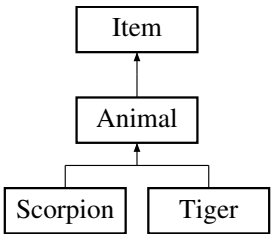
Animal.h . . . . . ??
Banana.h . . . . . ??
Coconut.h . . . . . ??
Collision.h . . . . . ??
Gamecontroller.h . . . . . ??
Gui.h . . . . . ??
Heart.h . . . . . ??
Item.h . . . . . ??
Main.h . . . . . ??
Menu.h . . . . . ??
Player.h . . . . . ??
Scorpion.h . . . . . ??
Tiger.h . . . . . ??
Tree.h . . . . . ??

Chapter 5

Class Documentation

5.1 Animal Class Reference

Inheritance diagram for Animal:



Public Member Functions

- **Animal** ()

  Construct a new Animal:: Animal object.
- virtual ~**Animal** ()=0

  Destroy the Animal:: Animal object.
- virtual void **move** (bool gamebreak)

  This function declares the direction of the object calls the function getDirection then gets an int value that tells in which direction the animal sould move.

Protected Member Functions

- int getDirection ()

  This function creates an random int number int number for up, down, left, right.

Additional Inherited Members

5.1.1 Member Function Documentation

5.1.1.1 getDirection()

```
int Animal::getDirection ( )  [protected]
```

This function creates an random int number int number for up, down, left, right.

Returns

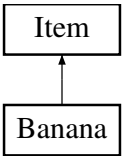
int between 1-4

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

- Animal.h
- Animal.cpp

5.2 Banana Class Reference

Inheritance diagram for Banana:



Public Member Functions

- Banana ()

  Construct a new Banana:: Banana object default constructor.
- ~**Banana** ()

  Destroy the Banana:: Banana object.

Additional Inherited Members

5.2.1 Constructor & Destructor Documentation

5.2.1.1 Banana()

```
Banana::Banana ( )
```

Construct a new Banana:: Banana object default constructor.

Construct a new Banana:: Banana object constructor with x and y values

Parameters

x	float x position of banana
y	float y position of banana

if file couldn`t load

Smooths the pixels (Sharpen the image)

Give Image to texture

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

- Banana.h
- Banana.cpp

5.3 Collision::BitmaskManager Class Reference

Public Member Functions

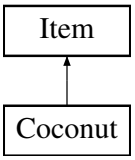
- sf::Uint8 **GetPixel** (const sf::Uint8 ∗mask, const sf::Texture ∗tex, unsigned int x, unsigned int y)
- sf::Uint8 ∗ **GetMask** (const sf::Texture ∗tex)
- sf::Uint8 ∗ **CreateMask** (const sf::Texture ∗tex, const sf::Image &img)

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

- Collision.cpp

5.4 Coconut Class Reference

Inheritance diagram for Coconut:



Public Member Functions

- **Coconut** ()

  Construct a new Coconut:: Coconut object default constructor of coconut.
- Coconut (float x, float y, int i)

  Construct a new Coconut:: Coconut object constructor with x and y values.
- ∼**Coconut** ()

  Destroy the Coconut:: Coconut object.
- void **flyCoconut** (bool gamebreak)

  this function let the coconuts fly

Additional Inherited Members

5.4.1 Constructor & Destructor Documentation

5.4.1.1 Coconut()

```
Coconut::Coconut (
            float x,
            float y,
            int i )
```

Construct a new Coconut:: Coconut object constructor with x and y values.

Parameters

x	float x position of Coconut
y	float y position of Coconut
i	i=1 brown Coconut, i=0 red Coconut

Smooths the pixels (Sharpen the image)

Give Image to texture

declarate sprite

Texture to Sprite

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

- Coconut.h
- Coconut.cpp

5.5 Gamecontroller Class Reference

Public Member Functions

- **Gamecontroller** ()

  Construct a new Gamecontroller:: Gamecontroller object default Constructor.
- ~**Gamecontroller** ()

  Destroy the Gamecontroller:: Gamecontroller object.
- void startGame ()

  start the Game intialise the music create Object of Gui call checkWindow()

5.5.1 Member Function Documentation

5.5.1.1 startGame()

```
void Gamecontroller::startGame ( )
```

start the Game intialise the music create Object of Gui call checkWindow()

initialise the music

load the music file

create Object of Gui

call checkWindow() to initialise Jungle, Home an Manpage

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

- Gamecontroller.h
- Gamecontroller.cpp

5.6 Gui Class Reference

Public Member Functions

- **Gui** ()
  Construct a new Gui:: Gui object call initVariables() and initWindow()
- const bool running ()
  checks if the window is open
- void **update** ()
  This function checks boarders and events for the objects.
- void **render** ()
  This function renders the window with a rgb-color, draws the shape and sprite and display it on the screen.
- void pollEvents ()
  This function checks if there is a Mouse or Keyboard event an the window proofe wasd or up, down, left, right proofe if window was closed.
- void checkWindow ()
  This function sets the setup for the "startwindow" and updates, renders window and proofe collision while its running.
- void **moveObjects** ()
  this function moves all Objects moves coconuts move scorpions moves Tiger
- void **resetClock** ()
  reset Clock for mooving objects
- void checkObjBorder (Item ∗object)
  this function checks if the passed Object will leave the window
- void **checkMyBorders** ()
  this function checks if any object will leave the window in the next step
- void createHome ()
  create Gamethings
- void createJungle ()
  this function initialise object jungle
- void openManpage ()
  Manpage.
- ∼**Gui** ()
  delete object Gui and this window
- void checkCollision ()
  changes for collision method

Public Attributes

- bool **gamebreak** =false
- int **counter** =0

5.6.1 Member Function Documentation

5.6.1.1 checkCollision()

```
void Gui::checkCollision ( )
```

changes for collision method

this function checks player and the other objects for collision check collision between Tiger and Player

check collision between Scorpion and Player

check collision between (brown) Coconut and Player

check collision between red Coconut and Player

check collision between heart and Player

check collision between Banana and Player

check collision between Tree and Player

check if the player is death of has won

5.6.1.2 checkObjBorder()

```
void Gui::checkObjBorder (
            Item * object )
```

this function checks if the passed Object will leave the window

Parameters

object	every Object of Item that moves in the window can be passed

Left

Right

Top

Bottom

5.6.1.3 checkWindow()

```
void Gui::checkWindow ( )
```

This function sets the setup for the "startwindow" and updates, renders window and proofe collision while its running.

set up the winow for the first time

this loop runs while the game is running

set the info menu

set the won or lost info

take new events from the gui

remains here if the game is on pause, info or finished

remains here if the game is running

moves the objects and check the border

reset the clock for the moving objects

check collision between Player and other objects

draw the objects on the window

check if the game is finished (death or won)

update the amount of bananas and heart in the Menubar

5.6.1.4 createHome()

```
void Gui::createHome ( )
```

create Gamethings

this function initialise objet home if couldn`t load image

Smooths the pixels (Sharpen the image)

Give Image to texture

5.6.1.5 createJungle()

```
void Gui::createJungle ( )
```

this function initialise object jungle

if couldn`t load image

Smooths the pixels (Sharpen the image)

Give Image to texture

5.6.1.6 openManpage()

```
void Gui::openManpage ( )
```

Manpage.

opens and reads txt-file "manpage"

5.6.1.7 pollEvents()

```
void Gui::pollEvents ( )
```

This function checks if there is a Mouse or Keyboard event an the window proofe wasd or up, down, left, right proofe if window was closed.

close window if red "X" is clicked

moves the Ape (player) with "wasd" or "arrow keys"

Left

Right

Up

Down

chekcks event if a button was clicked

if button play/pause button is clicked

if button info is clicked

5.6.1.8 running()

```
const bool Gui::running ( )
```

checks if the window is open

Returns

true if window is open

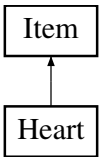
false if window is closed

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

- Gui.h
- Gui.cpp

5.7 Heart Class Reference

Inheritance diagram for Heart:



Public Member Functions

- Heart ()

  Construct a new Heart:: Heart object default constructor.
- ∼**Heart** ()

  Destroy the Heart:: Heart object.

Additional Inherited Members

5.7.1 Constructor & Destructor Documentation

5.7.1.1 Heart()

```
Heart::Heart ( )
```

Construct a new Heart:: Heart object default constructor.

Construct a new Heart:: Heart object constructor with x and y position

Parameters

x	float x position of heart
y	float y position of heart

if file couldn`t load

Smooths the pixels (Sharpen the image)

Give Image to texture

declarate Sprite

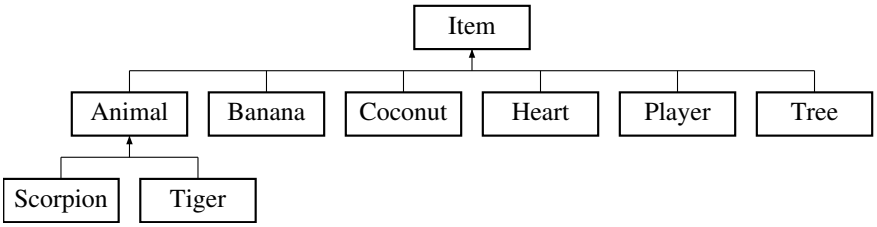
Texture to Sprite

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

- Heart.h
- Heart.cpp

5.8 Item Class Reference

Inheritance diagram for Item:



Public Member Functions

- **Item** ()

  Construct a new Item:: Item object.
- virtual ~**Item** ()=0

  Destroy the Item:: Item object.
- float setRandomPosX ()

  This function generates a random value for the x koordinate, which still is inside the gamefield.
- float setRandomPosY ()

  This function generates a random value for the y koordinate, which still is inside the gamefield.
- void **newPosition** ()

Public Attributes

- sf::Sprite **sprite**

  declare sprite
- float **itemWidth** = 20.f
- float **itemHeight** = 20.f
- sf::Clock **clock**
- sf::Time **framecounter** = sf::Time::Zero
- sf::Time **framecounterold** = sf::Time::Zero
- sf::Time **timePerFrame** = sf::seconds(1.f/60.f)

Protected Attributes

- float **posX** = 0
- float **posY** = 0
- sf::Image **image**

  create Item
- sf::Texture **texture**
- int **speed** = 100
- float **scaleWidth** = 0.1
- float **scaleHeight** = 0.1

5.8.1 Member Function Documentation

5.8.1.1 setRandomPosX()

```
float Item::setRandomPosX ( )
```

This function generates a random value for the x koordinate, which still is inside the gamefield.

Returns

float

5.8.1.2 setRandomPosY()

```
float Item::setRandomPosY ( )
```

This function generates a random value for the y koordinate, which still is inside the gamefield.

Returns

float

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

- Item.h
- Item.cpp

5.9 Menu Class Reference

Public Member Functions

- void checkInfoButton (int value)

  this method set the info Button depending of value
- void checkWonLostImage (int value)

  checks the won lost
- bool getInfo ()

  this method returns the Info value
- bool getPause ()

  this method return the pause value
- void pauseclicked ()
- void infoclicked ()
- void updateBananaHeart (Player *myPlayer)

  this method updates the amount of banana and Hearts in the menubar

Public Attributes

- sf::Texture **infoButton**
- sf::Sprite **infoButtonImage**
- sf::Texture **playButton**

  Play-Pause Button.
- sf::Sprite **playButtonImage**
- sf::Texture **menu**

  set menubar
- sf::Sprite **menuImage**
- sf::Text **infoText**
- sf::Texture **heart**

  set Heart picture
- sf::Sprite **heartImage**
- sf::Texture **banana**

  set Banana picture
- sf::Sprite **bananaImage**
- sf::Texture **infowindow**

  set infowindow
- sf::Sprite **infowindowImage**
- sf::Text **bananaText**
- sf::Text **amountBananaText**
- sf::Text **livesText**
- sf::Font **font**
- sf::Text **amountlivesText**
- sf::Text **wonGamesText**
- sf::Text **amountwonGamesText**

5.9.1 Member Function Documentation

5.9.1.1 checkInfoButton()

```
void Menu::checkInfoButton (
            int value )
```

this method set the info Button depending of value

Parameters

value	

if info menu

make info visible

if it is running

make info not visible

5.9.1.2 checkWonLostImage()

```
void Menu::checkWonLostImage (
            int value )
```

checks the won lost

Parameters

value	

won the game

make won info visible

make info not visible

5.9.1.3 getInfo()

```
bool Menu::getInfo ( )
```

this method returns the Info value

Returns

true : the game is on pause

false : the game is running

5.9.1.4 getPause()

```
bool Menu::getPause ( )
```

this method return the pause value

Returns

true : the game is on pause

false : the game is running

5.9.1.5 infoclicked()

```
void Menu::infoclicked ( )
```

set pause if info is opened and do not restart the gae automatically if info is closed --> have to press play to restart the game

5.9.1.6 pauseclicked()

```
void Menu::pauseclicked ( )
```

remove the info window if play was clicked

remove the info text

5.9.1.7 updateBananaHeart()

```
void Menu::updateBananaHeart (
            Player * myPlayer )
```

this method updates the amount of banana and Hearts in the menubar

Parameters

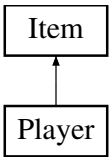
myPlayer	is the Object with amount of banana and heart

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

- Menu.h
- Menu.cpp

5.10 Player Class Reference

Inheritance diagram for Player:



Public Member Functions

- **Player** ()

  Construct a new Player:: Player object default constructor of Player.
- Player (float x, float y)

  Construct a new Player:: Player object constructor of Player with x and y variables.
- ∼**Player** ()

  Destroy the Player:: Player object.
- int getBanana ()

  this function return the amount of Bananas
- void **addBanana** ()

  this function increase banana if Player has collected one
- int get_Lives ()

this function returns the amount of Lives

- void set_Lives (int live)

  set new amount of lives increase if Player collected one decrease if hit scorpion or coconut
- int **getWon** ()
- bool checkWon ()

  This function checks if the player has enough (3) bananas collected.
- bool checkDeath ()

  This function checks if the player has more live than 0.
- int getIsfinished ()

  this method retuns the isfinished value
- void setIsfinished (int val)

  this method set the isfinished value
- void **set_Banana** (int banana)

  set the amounts of banana

Additional Inherited Members

5.10.1 Constructor & Destructor Documentation

5.10.1.1 Player()

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

Construct a new Player:: Player object constructor of Player with x and y variables.

Parameters

x	float x variable of Player
y	float y variable of Player

Load imageApe and if it couldn`t load

Smooths the pixels (Sharpen the image)

Give Image to texture

5.10.2 Member Function Documentation

5.10.2.1 checkDeath()

```
bool Player::checkDeath ( )
```

This function checks if the player has more live than 0.

Returns

true - if the player has lost all its lieves
false - if the player has enough live to play

5.10.2.2 checkWon()

```
bool Player::checkWon ( )
```

This function checks if the player has enough (3) bananas collected.

Returns

true - when player has enough collected banana
false - when player has to collect more banana

5.10.2.3 get_Lives()

```
int Player::get_Lives ( )
```

this function returns the amount of Lives

Returns

int amount of Lives

5.10.2.4 getBanana()

```
int Player::getBanana ( )
```

this function return the amount of Bananas

Returns

int amount of Bananas

5.10.2.5 getIsfinished()

```
int Player::getIsfinished ( )
```

this method retuns the isfinished value

Returns

int 0=game running, 1=won, 2=death

5.10.2.6 set_Lives()

```
void Player::set_Lives (
            int live )
```

set new amount of lives increase if Player collected one decrease if hit scorpion or coconut

Parameters

live	amount of live to increase/decrease

Returns

int amount of new lives

5.10.2.7 setIsfinished()

```
void Player::setIsfinished (
            int val )
```

this method set the isfinished value

Parameters

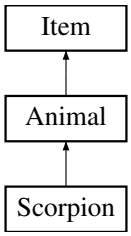
val	0=game running, 1=won, 2=death

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

- Player.h
- Player.cpp

5.11 Scorpion Class Reference

Inheritance diagram for Scorpion:



Public Member Functions

- Scorpion ()

  Construct a new Scorpion:: Scorpion object.
- ~**Scorpion** ()

  Destroy the Scorpion:: Scorpion object.

Additional Inherited Members

5.11.1 Constructor & Destructor Documentation

5.11.1.1 Scorpion()

```
Scorpion::Scorpion ( )
```

Construct a new Scorpion:: Scorpion object.

Parameters

x	float x position of Scorpion
y	float y position of Scorpion

Smooths the pixels (Sharpen the image)

Give Image to texture

declarate Sprite

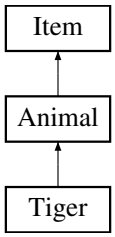
Texture to Sprite

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

- Scorpion.h
- Scorpion.cpp

5.12 Tiger Class Reference

Inheritance diagram for Tiger:



Public Member Functions

- Tiger ()

  Construct a new Tiger:: Tiger object.
- ∼**Tiger** ()

  Destroy the Tiger:: Tiger object.

Public Attributes

- const sf::Sprite **Object**

Additional Inherited Members

5.12.1 Constructor & Destructor Documentation

5.12.1.1 Tiger()

```
Tiger::Tiger ( )
```

Construct a new Tiger:: Tiger object.

Parameters

x	float x value of Tiger
y	float y value of Tiger

load immage and print warning if it couldn`t load

set correct sizes of object

Smooths the pixels (Sharpen the image)

Give Image to texture

declarate Sprite

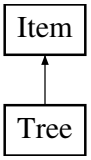
Texture to Sprite

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

- Tiger.h
- Tiger.cpp

5.13 Tree Class Reference

Inheritance diagram for Tree:



Public Member Functions

- Tree ()

  Construct a new Tree:: Tree object.
- void treeCollision (Player ∗player, Tree ∗tree)

  This Function blocks the tree on the Gamefield for the player.
- ∼**Tree** ()

  Destroy the Tree:: Tree object.

Additional Inherited Members

5.13.1 Constructor & Destructor Documentation

5.13.1.1 Tree()

```
Tree::Tree ( )
```

Construct a new Tree:: Tree object.

Smooths the pixels (Sharpen the image)

Give Image to texture

5.13.2 Member Function Documentation

5.13.2.1 treeCollision()

```
void Tree::treeCollision (
            Player * player,
            Tree * tree )
```

This Function blocks the tree on the Gamefield for the player.

Parameters

player	
tree	

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

- Tree.h
- Tree.cpp

Chapter 6

File Documentation

6.1 Animal.h

```
1 #ifndef Animal_h
2 #define Animal_h
3 #include <SFML/Graphics.hpp>
4 #include "Item.h"
5 #include <iostream>
6
7 class Animal: public Item
8 {
9 protected:
10     int getDirection();
11
12 public:
13     Animal();
14     virtual ~Animal() = 0;
15     virtual void move(bool gamebreak);
16 };
17
18
19 #endif
```

6.2 Banana.h

```
1 #pragma once
2 #include "Item.h"
3
4 class Banana: public Item
5 {
6 public:
7     Banana();
8     //Banana(float x, float y);
9     ~Banana();
10
11 };
```

6.3 Coconut.h

```
1 #pragma once
2 #include "Item.h"
3
4 class Coconut : public Item {
5 private:
6     int damage = 1;
7     int direction = 1;
8
9 public:
10     Coconut();
11     Coconut(float x, float y, int i);
12     ~Coconut();
13     void flyCoconut(bool gamebreak);
14
15
16 };
```

6.4 Collision.h

```
1 #ifndef COLLISION_H
2 #define COLLISION_H
3
4 namespace Collision {
5
6     bool PixelPerfectTest(const sf::Sprite& Object1 ,const sf::Sprite& Object2, sf::Uint8 AlphaLimit = 0);
7 }
8
9 #endif  /* COLLISION_H */
```

6.5 Gamecontroller.h

```
1 #ifndef Gamecontroller_h
2 #define Gamecontroller_h
3 #include "Player.h"
4 #include "Item.h"
5 #include "Banana.h"
6 #include "Heart.h"
7 #include "Gui.h"
8 #include "Animal.h"
9 #include "Tiger.h"
10 #include "Scorpion.h"
11 #include <iostream>
12 #include <stdlib.h>     /* srand, rand */
13 #include <time.h>       /* time */
14 #include <SFML/Audio.hpp>
15
16 class Gamecontroller
17 {
18 private:
19     int runtime = 0;
20
21 public:
22     Gamecontroller();
23     ~Gamecontroller();
24
25     void startGame();
26
27 };
28
29 #endif
```

6.6 Gui.h

```
1 #pragma once
2 #include <SFML/Graphics.hpp>
3 #include "Heart.h"
4 #include "Banana.h"
5 #include "Coconut.h"
6 #include "Tiger.h"
7 #include "Scorpion.h"
8 #include "Tree.h"
9 #include "Player.h"
10 #include "Collision.h"
11 #include "Menu.h"
12 #include <fstream>
13
14 class Gui
15 {
16 private:
17     sf::RenderWindow* window;
18     sf::Event event;
19     sf::VideoMode videoMode;
20     sf::Image icon;
21     sf::Text screenText;
22
23     // create Label
24     std::string txt_line;
25     sf::Font font_comic;
26     sf::Text txt_manpage;
27
29     Player *myPlayer = new Player(STARTPOSX, STARTPOSY);
30
32     Tiger *myTiger = new Tiger();
33
35     Scorpion *myScorpion = new Scorpion();
36     Scorpion *myScorpion2 = new Scorpion();
```

```
37      Scorpion *myScorpion3= new Scorpion();
38
40      Heart *myHeart = new Heart();
41
42
44      Coconut *myCoconut = new Coconut(-200.f,80.f, 1);
45      Coconut *myCoconut2 = new Coconut(-500.f,200.f, 1);
46      Coconut *myCoconut3 = new Coconut(-800.f,270.f, 1);
47      Coconut *myCoconut4 = new Coconut(-1000.f,320.f, 1);
48      Coconut *myCoconut5 = new Coconut(-1200.f,50.f, 0);
49
51      Tree *myTree = new Tree();
52      Tree *myTree2 = new Tree();
53      Tree *myTree3 = new Tree();
54      Tree *myTree4 = new Tree();
55      Tree *myTree5 = new Tree();
56
58      Banana *myBanana = new Banana();
59
61      sf::Image imageHome;
62      sf::Texture textureHome;
63      sf::RectangleShape shapeHome;
64      sf::Sprite spriteHome;
65
67      sf::Image imageJungle;
68      sf::Texture textureJungle;
69      sf::RectangleShape shapeJungle;
70
72      Menu *myMenu = new Menu();
73
74      int count=0;
75
76      void initVariables();
77      void initWindow();
78      void checkfinished();
79
80  public:
81      bool gamebreak=false;
82      int counter=0;
83      Gui();
84      const bool running();
85      void update();
86      void render();
87      void pollEvents();
88      void checkWindow();
89      void moveObjects();
90      void resetClock();
91      void checkObjBorder(Item *object);
92      void checkMyBorders();
94      void createHome();
95      void createJungle();
96
98      void openManpage();
99
100      ~Gui();
101
103      void checkCollision();
104 };
```

6.7 Heart.h

```
1  #pragma once
2  #include "Item.h"
3
4  class Heart: public Item
5  {
6  private:
7
8      int lifetime;
9  public:
10      Heart();
11      //Heart(float x, float y);
12      ~Heart();
13
14 };
```

6.8 Item.h

```
1  #pragma once
```

```
2 #include <SFML/Graphics.hpp>
3 #include <random>
4
5 //Randomizer
6 #define RANDOM std::random_device rdi; \
7     std::mt19937 geni(rdi()); \
8     std::uniform_real_distribution<> distX2(1,970); \
9     std::uniform_real_distribution<> distY2(50,572);
10
11 class Item
12 {
13 protected:
14     float posX = 0;
15     float posY = 0;
16
18     sf::Image image;
19     sf::Texture texture;
20
21     int speed = 100;
22     float scaleWidth = 0.1;
23     float scaleHeight = 0.1;
24
25 public:
26     Item();
27     virtual ~Item()=0;
28     //Random Values
29     float setRandomPosX();
30     float setRandomPosY();
31     void newPosition();
32
34     sf::Sprite sprite;
35     float itemWidth = 20.f;
36     float itemHeight = 20.f;
37
38     sf::Clock clock;
39     sf::Time framecounter = sf::Time::Zero;
40     sf::Time framecounterold = sf::Time::Zero;
41     sf::Time timePerFrame = sf::seconds(1.f/60.f);
42 };
```

6.9 Main.h

```
1 #ifndef Main_h
2 #define Main_h
3
4 #include <iostream>
5 #include <Windows.h>
6 #include "Gamecontroller.h"
7
8 #endif
```

6.10 Menu.h

```
1 #ifndef Menu_h
2 #define Menu_h
3 #include <SFML/Graphics.hpp>
4 #include <string>
5 #include <iostream>
6 #include <vector>
7 #include <fstream>
8 #include <sstream>
9 #include "Player.h"
10
11 using std::cout; using std::cin;
12 using std::endl; using std::string;
13 using std::to_string;
14 using std::cerr;
15 using std::ifstream; using std::ostringstream;
16
17 class Menu {
18     private:
19
20         bool info = false;
21         bool pause = false;
22         int amountBanana = 0;
23         int amountlives = 3;
24         int amountwongames =0;
25         string strbanana = "";
26         string strlives = "";
```

```
27          string strwon = "";
28
29
30          void setPauseButton();
31          void setPlayButton();
32          void setInfoButton();
33          void initialiseMenu();
34          string readFileIntoString(const string& path);
35
36
37      public:
38          sf::Texture infoButton;
39          sf::Sprite infoButtonImage;
40
42          sf::Texture playButton;
43          sf::Sprite playButtonImage;
44
46          sf::Texture menu;
47          sf::Sprite menuImage;
48
49          sf::Text infoText;
50
52          sf::Texture heart;
53          sf::Sprite heartImage;
54
56          sf::Texture banana;
57          sf::Sprite bananaImage;
58
60          sf::Texture infowindow;
61          sf::Sprite infowindowImage;
62
63          sf::Text bananaText;
64
65          sf::Text amountBananaText;
66
67          sf::Text livesText;
68
69          sf::Font font;
70
71
72          sf::Text amountlivesText;
73
74          sf::Text wonGamesText;
75
76          sf::Text amountwonGamesText;
77
78          Menu();
79          ~Menu();
80          void checkInfoButton(int value);
81          void checkWonLostImage(int value);
82          bool getInfo();
83          bool getPause();
84          void pauseclicked();
85          void infoclicked();
86          void updateBananaHeart(Player* myPlayer);
87
88 };
89 #endif
```

6.11 Player.h

```
1 #ifndef Player_h
2 #define Player_h
3 #include <SFML/Graphics.hpp>
4 #include "Item.h"
5
6 #define STARTPOSX 40.f
7 #define STARTPOSY 550.f
8
9 class Player: public Item
10 {
11 private:
12     int lives = 5;
13     int collected_banana = 0;
14     int won_games = 0;
15     int isfinished =0;  //0=running; 1=won; 2=death
16
17
18
19 public:
20     Player();
21     Player(float x, float y);
22     ~Player();
```

```
23
24      int getBanana();
25      void addBanana();
26      int get_Lives();
27      void set_Lives(int live);
28      int getWon();
29      bool checkWon();
30      bool checkDeath();
31      int getIsfinished();
32      void setIsfinished(int val);
33      void set_Banana(int banana);
34  };
35
36  #endif
```

6.12 Scorpion.h

```
1  #ifndef Scorpion_h
2  #define Scorpion_h
3  #include "Animal.h"
4
5  class Scorpion : public Animal
6  {
7  private:
8      int damage = 1;
9  public:
10      Scorpion();
11      //Scorpion(float x, float y);
12      ~Scorpion();
13
14  };
15
16  #endif
```

6.13 Tiger.h

```
1  #ifndef Tiger_h
2  #define Tiger_h
3  #include "Animal.h"
4
5  class Tiger : public Animal
6  {
7  private:
8      int damage = 5;
9      int lifetime = 0;
10
11  public:
12
13      Tiger();
14      //Tiger(float x, float y);
15      ~Tiger();
16      //changes
17      const sf::Sprite Object;
18  };
19
20  #endif
```

6.14 Tree.h

```
1  #ifndef Tree_h
2  #define Tree_h
3  #include "Item.h"
4  #include "Player.h"
5
6  class Tree: public Item
7  {
8
9  public:
10      Tree();
11      //Tree(float x, float y);
12
13      void treeCollision(Player *player, Tree *tree);
14
15      ~Tree();
16
```

```
17 };
18
19
20
21 #endif
```