Flappy Bird

1.0

Generated by Doxygen 1.9.1

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 Bird Class Reference

Main character of the game.

```
#include <Bird.hpp>
```

Public Member Functions

- Bird ()

  Constructor of Bird.
- void fly ()

  Fly command.
- void fall ()

  Fall command.
- void update ()

  Update command.

Public Attributes

- sf::Sprite body

  Body of the bird.

3.1.1 Detailed Description

Main character of the game.

Definition at line 7 of file Bird.hpp.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Bird()

```
Bird::Bird ( )
```

Constructor of Bird.

Initializes the bird in terms velocity, acceleration, and max flying position. Loads the bird image and sets its position, texture, and scale.

3.1.3 Member Function Documentation

3.1.3.1 fall()

```
void Bird::fall ( )
```

Fall command.

Sets the bird's acceleration to the predefined acceleration value to cause falling illusion.

3.1.3.2 fly()

```
void Bird::fly ( )
```

Fly command.

Sets the bird's acceleration to the predefined acceleration value to cause flying illusion. Computes the maximum vertical position the bird can reach according to its current position.

3.1.3.3 update()

```
void Bird::update ( )
```

Update command.

Updates the velocity of the bird and moves it accordingly. Min and max boundaries are defined to keep the velocity within a range. Evaluates the position of the bird and starts falling if applicable.

3.1.4 Member Data Documentation

3.1.4.1 body

```
sf::Sprite Bird::body
```

Body of the bird.

Definition at line 13 of file Bird.hpp.

3.2 Game Class Reference

Where game logic happens.

```
#include <Game.hpp>
```

Public Member Functions

- Game (const char ∗title)

  Constructor of the class Game.
- ∼Game ()

  Destructor of the class Game.
- void mainloop ()

  Loop game logic.

Public Attributes

- int score = 0

  Number of pipes the bird has passed.

3.2.1 Detailed Description

Where game logic happens.

Definition at line 11 of file Game.hpp.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Game()

```
Game::Game (
              const char ∗ title )
```

Constructor of the class Game.

Builds the window with a given title, loads the background image, sets up the background scale and texture, and starts the bird's movement to falling trajectory.

3.2.2.2 ∼Game()

```
Game::∼Game ( )
```

Destructor of the class Game.

Prints the final score.

3.2.3 Member Function Documentation

3.2.3.1 mainloop()

```
void Game::mainloop ( )
```

Loop game logic.

Whenever the window is open, checks for events, updates the logic of the game, and re-draws the window and the elements in it. Loop breaks when a collision between the bird and an object is detected.

3.2.4 Member Data Documentation

3.2.4.1 score

```
int Game::score = 0
```

Number of pipes the bird has passed.

Increses whenever the back of the bird passes the right side of a pair of an obstacle.

Definition at line 19 of file Game.hpp.

3.3 Obstacle Class Reference

Pair of pipes: top pipe faces down, bottom pipe faces up.

```
#include <Obstacle.hpp>
```

Public Member Functions

- Obstacle ()

    Constructor of the class Obstacle.
- void update ()

    Update the position of the obstacle.
- bool section ()

    Detect section edge.
- bool disappear ()

    Detect window edge.
- bool passed (int threshold)

    Detect threshold pass.

Public Attributes

- sf::Sprite top

    Top pipe of the obstacle.
- sf::Sprite bottom

    Bottom pipe of the obstacle.

3.3.1 Detailed Description

Pair of pipes: top pipe faces down, bottom pipe faces up.

Definition at line 8 of file Obstacle.hpp.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Obstacle()

```
Obstacle::Obstacle ( )
```

Constructor of the class Obstacle.

An obstacle is a pair of two pipes. The constructor loads the obstacle image for both top and bottom pipes, sets their position randomly within given boundaries, and sets the texture and scale of the pipes.

3.3.3 Member Function Documentation

3.3.3.1 disappear()

```
bool Obstacle::disappear ( )
```

Detect window edge.

Returns true whenever an obstacle reaches the left-most position so that it is not visible in the window anymore.

3.3.3.2 passed()

```
bool Obstacle::passed (
             int threshold )
```

Detect threshold pass.

Returns true whenever the obstacle passed a given position in the horizontal axis.

Parameters

threshold	Position to evaluate against

3.3.3.3 section()

```
bool Obstacle::section ( )
```

Detect section edge.

Returns true whenever an obstacle reaches a section edge, as per the pre-defined section dimensions.

3.3.3.4 update()

```
void Obstacle::update ( )
```

Update the position of the obstacle.

Moves both top and bottom pipes to the left, according to a pre-defined obstacle velocity.

3.3.4 Member Data Documentation

3.3.4.1 bottom

```
sf::Sprite Obstacle::bottom
```

Bottom pipe of the obstacle.

Definition at line 19 of file Obstacle.hpp.

3.3.4.2 top

```
sf::Sprite Obstacle::top
```

Top pipe of the obstacle.

Definition at line 14 of file Obstacle.hpp.

Chapter 4

File Documentation

4.1 include/Bird.hpp File Reference

```
#include "Constants.hpp"
#include <iostream>
```

Classes

- class Bird

  Main character of the game.

Macros

- #define FLY_MAX 10

  Bird's max position.

4.1.1 Macro Definition Documentation

4.1.1.1 FLY_MAX

```
#define FLY_MAX 10
```

Bird's max position.

Maximum vertical position a bird can fly to with respect to its original position.

Definition at line 102 of file Constants.hpp.

4.2 include/Constants.hpp File Reference

Macros

- #define WIN_WIDTH 800

  Window width.
- #define WIN_HEIGHT 1024

  Window height.
- #define WIN_SCALE_X 2.5f

  Scale factor for the backgound image in x-axis.
- #define WIN_SCALE_Y 2.0f

  Scale factor for the backgound image in y-axis.
- #define NUM_SECTIONS 3

  Number of sections.
- #define SECTION_GAP (WIN_WIDTH / NUM_SECTIONS)

  Width of each section.
- #define OBST_WIDTH 100

  Width of each obstacle.
- #define OBST_GAP 350

  Vertical space between the top and the bottom pipes.
- #define OBST_VEL -0.08f

  Velocity of the pipes moving to the left.
- #define GAP_MIN (2 ∗ OBST_GAP)

  Minimum allowed position of the obstacle gap on the vertical axis.
- #define GAP_MAX (WIN_HEIGHT - GAP_MIN)

  Maximum allowed position of the obstacle gap on the vertical axis.
- #define BIRD_SCALE 0.20f

  Bird scale.
- #define BIRD_RADIUS 20

  Bird radius.
- #define VEL_MAX 0.4f

  Bird's maximum falling velocity.
- #define VEL_MIN -0.5f

  Bird's minimum falling velocity.
- #define ACC_FLY -0.01f

  Bird's flying acceleration.
- #define ACC_FALL 0.001f

  Bird's falling acceleration.

4.2.1 Macro Definition Documentation

4.2.1.1 ACC_FALL

```
#define ACC_FALL 0.001f
```

Bird's falling acceleration.

Flying acceleration is higher than falling one in order to represent the effect of gravity

Definition at line 95 of file Constants.hpp.

4.2.1.2 ACC_FLY

```
#define ACC_FLY -0.01f
```

Bird's flying acceleration.

Flying acceleration is higher than falling one in order to represent the effect of gravity

Definition at line 88 of file Constants.hpp.

4.2.1.3 BIRD_RADIUS

```
#define BIRD_RADIUS 20
```

Bird radius.

Definition at line 71 of file Constants.hpp.

4.2.1.4 BIRD_SCALE

```
#define BIRD_SCALE 0.20f
```

Bird scale.

Definition at line 66 of file Constants.hpp.

4.2.1.5 GAP_MAX

```
#define GAP_MAX (WIN_HEIGHT - GAP_MIN)
```

Maximum allowed position of the obstacle gap on the vertical axis.

Definition at line 61 of file Constants.hpp.

4.2.1.6 GAP_MIN

```
#define GAP_MIN (2 * OBST_GAP)
```

Minimum allowed position of the obstacle gap on the vertical axis.

Definition at line 55 of file Constants.hpp.

4.2.1.7 NUM_SECTIONS

```
#define NUM_SECTIONS 3
```

Number of sections.

Corresponding to the maximum number of full pipes present in the screen. Divides the window in N vertical sections, used to determine the pipe flow.

Definition at line 28 of file Constants.hpp.

4.2.1.8 OBST_GAP

```
#define OBST_GAP 350
```

Vertical space between the top and the bottom pipes.

Definition at line 44 of file Constants.hpp.

4.2.1.9 OBST_VEL

```
#define OBST_VEL -0.08f
```

Velocity of the pipes moving to the left.

Definition at line 49 of file Constants.hpp.

4.2.1.10 OBST_WIDTH

```
#define OBST_WIDTH 100
```

Width of each obstacle.

Definition at line 38 of file Constants.hpp.

4.2.1.11 SECTION_GAP

```
#define SECTION_GAP (WIN_WIDTH / NUM_SECTIONS)
```

Width of each section.

Definition at line 33 of file Constants.hpp.

4.2.1.12 VEL_MAX

```
#define VEL_MAX 0.4f
```

Bird's maximum falling velocity.

Definition at line 76 of file Constants.hpp.

4.2.1.13 VEL_MIN

```
#define VEL_MIN -0.5f
```

Bird's minimum falling velocity.

Definition at line 81 of file Constants.hpp.

4.2.1.14 WIN_HEIGHT

```
#define WIN_HEIGHT 1024
```

Window height.

Definition at line 9 of file Constants.hpp.

4.2.1.15 WIN_SCALE_X

```
#define WIN_SCALE_X 2.5f
```

Scale factor for the backgound image in x-axis.

Definition at line 14 of file Constants.hpp.

4.2.1.16 WIN_SCALE_Y

```
#define WIN_SCALE_Y 2.0f
```

Scale factor for the backgound image in y-axis.

Definition at line 19 of file Constants.hpp.

4.2.1.17 WIN_WIDTH

```
#define WIN_WIDTH 800
```

Window width.

Definition at line 4 of file Constants.hpp.

4.3 include/Game.hpp File Reference

```
#include "../src/Platform/Platform.hpp"
#include "Bird.hpp"
#include "Constants.hpp"
#include "Obstacle.hpp"
#include <iostream>
#include <vector>
```

Classes

- class Game

    Where game logic happens.

4.4 include/Obstacle.hpp File Reference

```
#include "Constants.hpp"
#include <iostream>
```

Classes

- class Obstacle

    Pair of pipes: top pipe faces down, bottom pipe faces up.