Angry Birds 10

Generated by Doxygen 1.9.1

Chapter 1

README

Instructions for using/making level files!

File format:

- '#' marks a comment (has to be at start of line)
  - # e.g. comment here
- Basic idea is Classname/defaultsName, x, y, rotation
  - x, y is the center of the object
  - rotation (optional) in degrees
    - object is rotated around center
- File is case-insensitive
- Extra spaces, extra text (at end of line), empty lines and bad lines will be ignored
- Level requires at least:
  - 1 bird
  - 1 pig
  - Realistically some blocks
    - To simplify level creation, a fixed ground will be automatically added below y=0 from x=-100 to x=100 (m)

Adding objects:

- Birds:
  - Classname
    - List of birds that can be shot, in this order
    - Whether/how to display unshot birds is left to rendering
    - Them being physics objects before launch could lead to unusual behaviour (e.g. falling off the map)
- Pigs:
  - NormalPig, x, y, rotation
  - IronPig, x, y

  - etc., name from default values

- Blocks:
  - DefaultsName, x, y, rotation
  - e.g.:
  - IceCircleS, x, y, rot
  - WoodSquare, x, y
  - etc.

Settings format:

- Given a setting multiple times the last one will be selected
- Gravity, x, y
  - Default is 0, -10 if omitted
- Slingshot, x, y
  - Default is -15, 2 if omitted
- ScoreLimits, 1∗, 2∗, 3∗
  - Score limits for stars
  - Defaults are <*reasonable defaults*> if omitted
  - Recommended to set these

Chapter 2

Meeting Notes

In this file, you are required to take notes for your weekly meetings. In each meeting, you are required to discuss:

1. What each member has done during the week?
2. Are there challenges or problems? Discuss the possible solutions
3. Plan for the next week for everyone
4. Deviations and changes to the project plan, if any

2.1 Meeting dd.mm.yy hh:mm

Participants:

1. Rautapää Jaakko
2. Amini Yalda
3. Zharkynuly Daniyar
4. Zambelly Soma
5. Xin Lin

2.1.1 Summary of works

1. Member 1

   Implementing the class XX. Tested the class XX. Results are in `tests/<class-xx-tests>`. Resolved the identified problems.

2. Member 2

   Same as above

3. ...

2.1.2 Challenges

1. The integration of UI with the monsters requires an abstract interface.
2. ...

2.1.3 Actions

1. Member 1 is going to look into defining an abstract interface for monsters to enable easy UI integration.
2. Member 2 is going to work with Member 1 to use abstract interface in derived monster classes.
3. Member 2 is going to test the interface.
4. Member 3 is going to use ...

   Please reflect these action decisions in your git commit messages so that your group members and advisor can follow the progress.

2.1.4 Project status

Short summary of current project status.

2.1.4.1 TODOs

1. Member 1: Write an action.
2. ...

2.2 Meeting 25.10.2024 14:15-15:30

Participants:

1. Rautapää Jaakko
2. Amini Yalda
3. Zharkynuly Daniyar
4. Zambelly Soma
5. Xin Lin

2.2.1 Summary of works

Set up repository, local clone, project plan template

2.2.2 Challenges

It feels a bit messy intially, and it took quite some time to sort out things together and get started. The main reason is that there is no clear meeting goal set, or preparation work done before the meeting.

2.2.3 Actions

All members are going to do some independant research and discuss ideas in next week's meeting.

2.2.4 Project status

Getting started. Now working on the project plan.

Chapter 3

Contents

Project plan is a PDF document describing the scope of the project, major architectural decisions, preliminary schedule and distribution of roles in the group, design rationale and so on. The document should be roughly five pages long, with a couple of diagrams illustrating the program design (for example, the planned class relationships).

You are required commit your project plan in this folder before the deadline. The plan should contain the following information:

- Scope of the work: what features and functionalities will be implemented, how is the program used, and how does it work
- High-level structure of the software: main modules, main classes (according to current understanding)
- Planned use of external libraries
- Division of work and responsibilities between the group
- Planned schedule and milestones before the final deadline of the project

  It is not uncommon that as the project progresses, there may be changes relative to project plan, and that is fine. The final outcome will be described in the final documentation, that can be based on the project plan.

Chapter 4

Angry Birds

This is the template for the projects. Please copy the project description here. You can use Markdown language to render it as formatted **HTML** file.

The player hurls birds or similar objects towards a fortress to destroy enemy targets. Player should use as few throws as possible for the higher score. Fortresses are built of destroyable and non-destroyable construction elements. Failing, i.e not destroying all of the enemies, typically means the level is reset and all points are lost.

The game has very simple rules, but as in any well-crafted game, the player's points should correlate with their throwing skills. This project subject does not require you to make a clone of an existing game, but to implement a one with the same general idea.

4.1 Group

- Jaakko Rautapää
- Xin Lin

4.2 Repository organization

Your project implementation should follow the skelaton organization in this repository. See readme.md files in each folder.

4.3 Project Implementation

You must use git repository for the work on the project, making frequent enough commits so that the project group (and course staff) can follow the progress.

The completed project work will be demonstrated to the group's advisor at a demo session. The final demonstrations are arranged on week 50. After the final demonstrations project group evaluates another project, and self-evaluates own project. In addition, project members will give a confidential individual assessment of each group member

The course staff should be able to easily compile the project work using makefile and related instructions provided in the git repository. The final output should be in the **master branch** of the git repository.

4.4 Working practices

Each project group is assigned an advisor from the project teaching personnel. There will be a dedicated Teams channel for each project topic to facilitate discussion between the groups in the same topic and the advisor.

The group should meet weekly. The weekly meeting does not need to be long if there are no special issues to discuss, and can be taken remotely as voice/video chat on the group Teams channel (or Zoom or other similar tool), preferably at a regular weekly time. In the meeting the group updates:

- What each member has done during the week
- Are there challenges or problems? Discuss the possible solutions
- Plan for the next week for everyone
- Deviations and changes to the project plan, if any
- After the meetings, the meeting notes will be committed to the project repository in the `Meeting-notes.md` file.
  - The commits within the week should have some commit messages referring to the meeting notes so that the project advisor can follow the progress.
  - **The meeting notes should be in English.**

  Everyone may not be able to participate to all meetings, but at least a couple of members should be present in each meeting. Regular absence from meetings will affect in individual evaluation.

4.5 Source code documentation

It is strongly recommended to use Doxygen to document your source code. Please go over the *Project Guidelines* for details.

4.6 TODOs (Date)

You can create a list of TODOs in this file. The recommended format is:

- Complete class implementation **foo**. Assigned to <Member 1>
- Test ...

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

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

Chapter 6

Class Index

6.1 Class List

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

Chapter 7

File Index

7.1 File List

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

Chapter 8

Class Documentation

8.1 Bird Class Reference

Abstract physics bird class.

`#include <bird.hpp>`

Inheritance diagram for Bird:

8.2 Block Class Reference

Inheritance diagram for Block:

Collaboration diagram for Block:

Public Member Functions

- **Block** (b2World *world, float x, float y, Defs *defaults, float rotation=0.0f)
- virtual void updateTexture (float deltaTime) override

  Delete object in timer_s seconds (deleted by Level).

Additional Inherited Members

8.2.1 Member Function Documentation

8.2.1.1 updateTexture()

```
virtual void Block::updateTexture (
            float deltaTime )  [inline], [override], [virtual]
```

Delete object in timer_s seconds (deleted by Level).

Objects need to be deleted between steps

and sounds take time to finish.

Parameters

timer_s	time to deletion

Implements Object.

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

- src/objects/block.hpp

8.3 Button Class Reference

Visual button class.

`#include <button.hpp>`

Public Member Functions

- **Button** (const sf::Texture &texture)
- void **updateSize** (float scaleX, float scaleY)
- void **setDefaultPosition** (float x, float y)
- void **updatePosition** (float scaleX, float scaleY)
- sf::Vector2f **getPosition** () const
- void **draw** (sf::RenderWindow &window)
- bool **isClicked** (const sf::Vector2f &mousePos) const
- void **activate** ()
- void **deactivate** ()
- bool **isActive** () const

8.3.1 Detailed Description

Visual button class.

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

- src/visual/button.hpp

8.4 ExplodeBird Class Reference

Inheritance diagram for ExplodeBird:

Collaboration diagram for ExplodeBird:

Public Member Functions

- **ExplodeBird** (b2World *world, float x, float y, bool flag=true)
- void Attack () override

  Bird does a special attack.
- virtual void updateTexture (float deltaTime) override

  Delete object in timer_s seconds (deleted by Level).
- virtual void **setDestroyTexture** () override

Protected Attributes

- int **blastRays** = 32
- float **blastRadius** = 5.0f
- float **blastPower** = 1000.0f
- size_t **row** = 0
- size_t **column** = 0
- std::vector< std::vector< size_t > > **texture_order** = {{0,1}, {0,2}, {0,3}, {0,4}}

Additional Inherited Members

8.4.1 Member Function Documentation

8.4.1.1 updateTexture()

```
virtual void ExplodeBird::updateTexture (
            float deltaTime )  [inline], [override], [virtual]
```

Delete object in timer_s seconds (deleted by Level).

Objects need to be deleted between steps

and sounds take time to finish.

Parameters

timer_s	time to deletion

Implements Object.

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

- src/objects/special_birds.hpp

8.5 Game Class Reference

Game class for the game.

`#include <game.hpp>`

Public Member Functions

- void run ()
  Runs the game.
- void update (float deltaTime)
  Updates the game.
- void draw ()
  Draws game and gui.
- void handleEvents ()
  Handles sfml events.
- b2Vec2 screenToWorldPos (const sf::Vector2i &screenPos)
  Transforms screen position to world position.
- void handleMousePress ()
  Handles mouse presses, performs actions based on it.
- void handleMouseRelease ()
  Handles mouse releases and performs relevant actions.
- void handleMouseMove ()
  Handles mouse movement and performs relevant actions.
- void handleButtonClicks (const std::string &button_name)
  Handles button clicks, clicks the button and performs it's action.
- b2World & **getWorld** ()
- void setLevel (int level_number)
  Sets and loads a level.
- void centerWindow ()
  Resizes and centers the window.

8.5.1 Detailed Description

Game class for the game.

8.5.2 Member Function Documentation

8.5.2.1 handleButtonClicks()

```
void Game::handleButtonClicks (
            const std::string & button_name ) [inline]
```

Handles button clicks, clicks the button and performs it's action.

Parameters

button_name	Name of button clicked (see gui)

8.5.2.2 screenToWorldPos()

```
b2Vec2 Game::screenToWorldPos (
            const sf::Vector2i & screenPos )  [inline]
```

Transforms screen position to world position.

Parameters

screenPos	to transform

Returns

b2Vec2 game pos

8.5.2.3 setLevel()

```
void Game::setLevel (
            int level_number )  [inline]
```

Sets and loads a level.

Parameters

level_number	level to load

8.5.2.4 update()

```
void Game::update (
            float deltaTime )  [inline]
```

Updates the game.

Parameters

deltaTime	update by this time (s)

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

- src/game.hpp

8.6 GameRender Class Reference

Renders the game on screen.

`#include <game_render.hpp>`

Public Member Functions

- **GameRender** (sf::RenderWindow &game_window)
- void setBounds ()

  Sets game bounds based on window size.
- bool inBounds (Object *object)

  Checks if an object is in bounds.
- void renderGame (Level &level)

  Renders the level on screen.
- void **setXBounds** (b2Vec2 bounds)
- void **setXBounds** (float x, float y)
- void **setYBounds** (b2Vec2 bounds)
- void **setYBounds** (float x, float y)
- b2Vec2 **getXBounds** ()
- b2Vec2 **getYBounds** ()
- b2Vec2 **getCenter** ()
- void setCenter (b2Vec2 center)

  Sets the center of game bounds such that bounds size stays the same.
- void **setCenter** (float x, float y)
- sf::Vector2f toScreenPos (const b2Vec2 &gamePos) const

  Transforms a game position to screen position. Considers both bounds.
- b2Vec2 toGamePos (const sf::Vector2f &screenPos) const

  Transforms a screen position to game position. Considers both bounds.

8.6.1 Detailed Description

Renders the game on screen.

8.6.2 Member Function Documentation

8.6.2.1 inBounds()

```
bool GameRender::inBounds (
            Object * object )  [inline]
```

Checks if an object is in bounds.

Parameters

object	

Returns

true if in bounds

8.6.2.2 renderGame()

```
void GameRender::renderGame (
            Level & level )  [inline]
```

Renders the level on screen.

Parameters

level	render this level

8.6.2.3 setCenter()

```
void GameRender::setCenter (
            b2Vec2 center )  [inline]
```

Sets the center of game bounds such that bounds size stays the same.

Parameters

center	new center point

8.6.2.4 toGamePos()

```
b2Vec2 GameRender::toGamePos (
            const sf::Vector2f & screenPos ) const  [inline]
```

Transforms a screen position to game position. Considers both bounds.

Parameters

screenPos	transform this

Returns

b2Vec2 game position

8.6.2.5 toScreenPos()

```
sf::Vector2f GameRender::toScreenPos (
            const b2Vec2 & gamePos ) const  [inline]
```

Transforms a game position to screen position. Considers both bounds.

Parameters

gamePos	transform this

Returns

sf::Vector2f screen pos

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

- src/visual/game_render.hpp

8.7 GameText Class Reference

Represents a visual text object.

```
#include <game_text.hpp>
```

Public Member Functions

- **GameText** (sf::Font font, int size, const sf::Color &color, const sf::Color &outlineColor, const sf::Vector2f &position, const std::string &text)
- **GameText** (int size, const sf::Vector2f &position, const std::string &text, const sf::Color &fillColor=sf::Color↩ ::White, const sf::Color &outlineColor=sf::Color::Black)
- void **setString** (const std::string &text)
- void **setDefaultPosition** (float x, float y)
- void **updatePosition** (float scaleX, float scaleY)
- void **setColor** (const sf::Color &color)
- void **setSize** (size_t size)
- void **updateSize** (float scaleX, float scaleY)
- void **draw** (sf::RenderWindow &window)

Static Public Member Functions

- static void **setDefaultFont** (sf::Font ∗font)

8.7.1 Detailed Description

Represents a visual text object.

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

- src/visual/game_text.hpp

8.8 Ground Class Reference

Public Member Functions

- **Ground** (b2World ∗world, const std::vector< b2Vec2 > &vertices)
- std::vector< b2Vec2 > & **getVertices** ()

Protected Attributes

- std::vector< b2Vec2 > **m_vertices**
- b2Body ∗ **body**

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

- src/objects/ground.hpp

8.9 GUI Class Reference

GUI class for the game.

`#include <gui.hpp>`

Public Member Functions

- **GUI** (sf::RenderWindow &game_window)
- void init ()

  Sets up the GUI.
- std::optional< std::string > getClickedButton (const sf::Vector2f &mousePos)

  Get name of button pos is on.
- void updateScale ()

  Updates the scale of the GUI based on window size.
- void updateAllPositions ()

  Update positions of all GUI elements.
- void updateAllSizes ()

  Update sizes of all GUI elements.
- void updateScore (int score)

  Update score texts.
- void updateBirdsLeft (int count)

Update birds left text.

- void **toggleMusic** ()
- void drawHome ()

  Draws the home screen.
- void drawHelp ()

  Draw help screen.
- void drawLevel ()

  Draw level screen.
- void drawGame (int level)

  Draws the ingame screen.
- void drawWin (int starCount)

  Draw win screen.
- void drawLost ()

  Draw lose screen.

8.9.1 Detailed Description

GUI class for the game.

8.9.2 Member Function Documentation

8.9.2.1 drawGame()

```
void GUI::drawGame (
            int level )  [inline]
```

Draws the ingame screen.

Parameters

level	level number

8.9.2.2 drawWin()

```
void GUI::drawWin (
            int starCount )  [inline]
```

Draw win screen.

Parameters

starCount	amount of stars

8.9.2.3 getClickedButton()

```
std::optional<std::string> GUI::getClickedButton (
            const sf::Vector2f & mousePos )  [inline]
```

Get name of button pos is on.

Parameters

mousePos	

Returns

std::optional<std::string> name of button

8.9.2.4 updateBirdsLeft()

```
void GUI::updateBirdsLeft (
            int count )  [inline]
```

Update birds left text.

Parameters

count	

8.9.2.5 updateScore()

```
void GUI::updateScore (
            int score )  [inline]
```

Update score texts.

Parameters

score	

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

- src/visual/gui.hpp

8.10 Level Class Reference

Represents a level.

`#include <level.hpp>`

Collaboration diagram for Level:

Public Member Functions

- **Level** (const std::string &levelPath, float frameRate=60.0f)
- **Level** (int level, float frameRate=60.0f)
- void loadLevel (const std::string path)

  Load a level from a level file.
- void **loadLevel** (int level)
- void update (float deltaTime)

  Updates the game.
- void addToDestroyList ()

  Adds objects to destroy list if necessary.
- void disableAndDestroy ()

  Disables and destroys objects in destroy list.
- void updateAllTexture ()

  Updates all textures.
- bool isWin ()

  Tells if game is won.
- bool isLost ()

  Tells if game is lost.
- int getStars ()

  Get number of stars based on score and score limits.
- std::vector< std::unique_ptr< Ground > > & **getGrounds** ()
- bool isMouseOnBird (const b2Vec2 &worldPos) const

  Is position on current active bird?
- bool startDragging (const b2Vec2 &worldPos)

  Try to start dragging bird in slingshot.
- void updateDragging (const b2Vec2 &worldPos)

  Updates bird dragging.
- void endDragging ()

  Ends bird dragging, launches bird.
- void setNextBird ()

  Set the next bird into the slingshot if available.
- b2World & **getWorld** ()
- Slingshot & **getSlingshot** ()
- Bird * **getCurrentBird** ()
- void **setActive** (bool active)
- bool **getActive** ()
- bool **getDragging** ()
- b2Vec2 **getGravity** ()
- float **getScore** ()
- void **setScore** (float value)
- void **addScore** (float add)
- float **getTimestep** ()
- const std::vector< Bird * > & **getBirds** ()
- const std::vector< Block * > & **getBlocks** ()
- const std::vector< Pig * > & **getPigs** ()
- const std::queue< std::string > & **getUnusedBirds** ()

Protected Member Functions

- void findErase (Object *toDelete)

  Finds object in birds, blocks or pigs and deletes it from the list.
- void clearLevel ()

  Clears and deletes level objects.
- const std::string getFilePath (int level)

  Level number to default file path.
- void parseLevelFile (std::ifstream &file)

  Parses and loads level file.
- void parseLine (std::stringstream &lineStream)

  Parses a line from file, performs action if valid line.
- bool addBird (const std::string &className)

  Adds unused bird based on name.
- bool addPig (const std::string &pigName, float x, float y, float rotation)

  Adds a pig to world.
- bool addBlock (const std::string &blockName, float x, float y, float rotation)

  Adds a block to world.
- bool setSetting (const std::string &setting, float x, float y, float z=-1)

  Sets a setting value.
- bool addGround (const std::string &parameter, std::stringstream &lineStream)

  Adds a ground to world.
- void addGroundBlocks ()

  Adds an invisible fixed ground to world.
- char asciitolower (char in)

  Character to lowercase.
- void toLower (std::string &string)

  Transforms string to lowercase, in place.
- float readFloat (std::stringstream &line)

  Reads a float from the stream, including a ',', the delimiter.

Protected Attributes

- Bird * **currentBird** = nullptr
- bool **isDragging** = false
- float **accumulator** = 0
- float **frameRate**
- float **timeStep**
- bool **isActive** = false
- float **scorePerUnusedBird** = 200.0f
- b2Vec2 **gravity**
- b2World **world**
- int32 **velocityIterations** = 6
- int32 **positionIterations** = 2
- std::queue< std::string > **unusedBirds**
- std::array< float, 3 > **scoreLimits**
- float **score** = 0
- std::vector< b2Vec2 > **groundPoints**
- std::vector< Bird * > **birds**
- std::vector< Pig * > **pigs**
- std::vector< Block * > **blocks**
- std::vector< std::unique_ptr< Ground > > **grounds**
- Slingshot **slingshot**
- ObjectCollisions **collisionHandler**

8.10.1 Detailed Description

Represents a level.

8.10.2 Member Function Documentation

8.10.2.1 addBird()

```
bool Level::addBird (
            const std::string & className )  [inline], [protected]
```

Adds unused bird based on name.

Parameters

className	name of bird class

Returns

true if succesful

8.10.2.2 addBlock()

```
bool Level::addBlock (
            const std::string & blockName,
            float x,
            float y,
            float rotation )  [inline], [protected]
```

Adds a block to world.

Parameters

blockName	block defaults name (see block)
x	world x pos
y	world y pos
rotation	in degrees

Returns

true if succesful

8.10.2.3 addGround()

```
bool Level::addGround (
            const std::string & parameter,
            std::stringstream & lineStream )  [inline], [protected]
```

Adds a ground to world.

Parameters

parameter	parameter from line
lineStream	line to read

Returns

true if succesful

8.10.2.4 addPig()

```
bool Level::addPig (
            const std::string & pigName,
            float x,
            float y,
            float rotation )  [inline], [protected]
```

Adds a pig to world.

Parameters

pigName	pig defaults name (see pig)
x	world x pos
y	world y pos
rotation	in degrees

Returns

true if succesful

8.10.2.5 asciitolower()

```
char Level::asciitolower (
            char in )  [inline], [protected]
```

Character to lowercase.

Parameters

in	char

Returns

char lowercase'd

8.10.2.6 findErase()

```
void Level::findErase (
            Object * toDelete )  [inline], [protected]
```

Finds object in birds, blocks or pigs and deletes it from the list.

Parameters

toDelete	remove this from the lists

8.10.2.7 getFilePath()

```
const std::string Level::getFilePath (
            int level )  [inline], [protected]
```

Level number to default file path.

Parameters

level	level number

Returns

const std::string level file path

8.10.2.8 getStars()

```
int Level::getStars ( )  [inline]
```

Get number of stars based on score and score limits.

Returns

int stars

8.10.2.9 isLost()

```
bool Level::isLost ( )  [inline]
```

Tells if game is lost.

Returns

true if lost

8.10.2.10 isMouseOnBird()

```
bool Level::isMouseOnBird (
            const b2Vec2 & worldPos ) const  [inline]
```

Is position on current active bird?

Parameters

worldPos	position in world coordinates

Returns

true if position on bird

8.10.2.11 isWin()

```
bool Level::isWin ( )  [inline]
```

Tells if game is won.

Returns

true if won

8.10.2.12 loadLevel()

```
void Level::loadLevel (
            const std::string path )  [inline]
```

Load a level from a level file.

Parameters

path	path to level file

8.10.2.13 parseLevelFile()

```
void Level::parseLevelFile (
            std::ifstream & file )  [inline], [protected]
```

Parses and loads level file.

Parameters

file	file object

8.10.2.14 parseLine()

```
void Level::parseLine (
            std::stringstream & lineStream )  [inline], [protected]
```

Parses a line from file, performs action if valid line.

Parameters

lineStream	line to parse

8.10.2.15 readFloat()

```
float Level::readFloat (
            std::stringstream & line )  [inline], [protected]
```

Reads a float from the stream, including a ',', the delimiter.

Parameters

line	the stream

Returns

the read float, or FLT_MIN if unsuccesful

8.10.2.16 setSetting()

```
bool Level::setSetting (
            const std::string & setting,
            float x,
            float y,
            float z = -1 )  [inline], [protected]
```

Sets a setting value.

Parameters

setting	Setting name
x	parameter 1
y	parameter 2
z	parameter 3

Returns

true if succesful

8.10.2.17 startDragging()

```
bool Level::startDragging (
            const b2Vec2 & worldPos )  [inline]
```

Try to start dragging bird in slingshot.

Parameters

worldPos	Drag from here

Returns

true if succesful

8.10.2.18 toLower()

```
void Level::toLower (
            std::string & string )  [inline], [protected]
```

Transforms string to lowercase, in place.

Parameters

string	to transform

8.10.2.19 update()

```
void Level::update (
            float deltaTime )  [inline]
```

Updates the game.

Parameters

deltaTime	update by this time

8.10.2.20 updateDragging()

```
void Level::updateDragging (
            const b2Vec2 & worldPos )  [inline]
```

Updates bird dragging.

Parameters

worldPos	Drag here

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

- src/level.hpp

8.11 NormalBird Class Reference

Physics bird with a special attack (not this one)

`#include <special_birds.hpp>`

Inheritance diagram for NormalBird:

Collaboration diagram for NormalBird:

Public Member Functions

- **NormalBird** (b2World *world, float x, float y)
- void Attack () override

  Bird does a special attack.
- virtual void updateTexture (float deltaTime) override

  Delete object in timer_s seconds (deleted by Level).
- virtual void **setDestroyTexture** () override

Additional Inherited Members

8.11.1 Detailed Description

Physics bird with a special attack (not this one)

8.11.2 Member Function Documentation

8.11.2.1 updateTexture()

```
virtual void NormalBird::updateTexture (
            float deltaTime )  [inline], [override], [virtual]
```

Delete object in timer_s seconds (deleted by Level).

Objects need to be deleted between steps

and sounds take time to finish.

Parameters

timer_s	time to deletion

Implements Object.

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

- src/objects/special_birds.hpp

8.12 Object Class Reference

Represents a basic physics object.

`#include <object.hpp>`

Inheritance diagram for Object:

Public Member Functions

- Object (b2World *world, b2BodyDef *bodyDef, b2Shape *shape, float density, float x, float y, float hp, std::vector< std::pair< std::string, float >> textureDefs, std::vector< std::pair< std::string, float >> damageTextureDefs, std::vector< std::string > destroySoundNames, std::vector< std::string > collisionSoundNames, std::vector< std::string > damageSoundNames, std::vector< std::string > otherSoundNames, float rotation=0.0f)

  Main constructor.
- Object (b2World *world, float x, float y, ObjectDefs::ObjectDefaults *defaults, float rotation=0.0f)

  Constructor used by derived classes.
- void loadSounds (std::vector< std::string > destroySoundNames, std::vector< std::string > collisionSoundNames, std::vector< std::string > damageSoundNames, std::vector< std::string > otherSoundNames)

  Loads sounds.
- void stopSounds ()

  Stops all sounds.
- bool playSound (const std::string &name)

  Plays a sound by name.
- bool **playSound** (soundType sound_type)
- void Destroy (float timer_s=0.f)

  Sets object to be deleted.
- float transferScore ()

  Gives score if destroyed, resets.
- virtual bool TakeDamage (float dmg)

  Takes damage, plays sound.
- virtual void **checkDamage** ()
- virtual void **setDestroyTexture** ()
- virtual void updateTexture (float deltaTime)=0

  Delete object in timer_s seconds (deleted by Level).
- b2Body * **getBody** ()
- float **getScore** ()
- void **setScore** (float newScore)
- float **getMaxHP** ()
- float **getHP** ()
- sf::Sprite & **getSprite** ()
- bool **getDisableOnDestroy** ()
- bool **getHasDestroyTexture** ()

Static Public Attributes

- constexpr static const float **speedDamageMultiplier** = 7.0f
- static std::list< std::pair< float, Object * > > **destroyList**

Protected Attributes

- b2Body * **body**
- sf::Sprite **sprite**
- std::vector< std::pair< std::string, float > > **normalTextures**
- std::vector< std::pair< std::string, float > > **damageTextures**
- std::vector< sf::Sound > **damageSounds**
- std::vector< sf::Sound > **collisionSounds**
- std::vector< sf::Sound > **destroySounds**
- std::map< std::string, sf::Sound > **otherSoundsMap**

- float **MaxHP**
- float **CurrentHP**
- float **score** = 0
- size_t **currentTextureIdx** = 0
- float **animationTimer** = 0.0f
- bool **isDamaged** = false
- bool **toBeDeleted** = false
- bool **disableOnDestroy** = true
- bool **hasDestroyTexture** = false

8.12.1 Detailed Description

Represents a basic physics object.

8.12.2 Constructor & Destructor Documentation

8.12.2.1 Object()

```
Object::Object (
            b2World * world,
            b2BodyDef * bodyDef,
            b2Shape * shape,
            float density,
            float x,
            float y,
            float hp,
            std::vector< std::pair< std::string, float >> textureDefs,
            std::vector< std::pair< std::string, float >> damageTextureDefs,
            std::vector< std::string > destroySoundNames,
            std::vector< std::string > collisionSoundNames,
            std::vector< std::string > damageSoundNames,
            std::vector< std::string > otherSoundNames,
            float rotation = 0.0f )  [inline]
```

Main constructor.

Parameters

world	World to add in
bodyDef	body def
shape	collider
density	
x,y	world pos
hp	max HP
textureDefs	normal textures
damageTextureDefs	damage textures
destroySoundNames	
collisionSoundNames	
damageSoundNames	
otherSoundNames	
rotation	in degrees

8.12.3 Member Function Documentation

8.12.3.1 Destroy()

```
void Object::Destroy (
            float timer_s = 0.f )  [inline]
```

Sets object to be deleted.

Parameters

timer_s	time to deletion

8.12.3.2 loadSounds()

```
void Object::loadSounds (
            std::vector< std::string > destroySoundNames,
            std::vector< std::string > collisionSoundNames,
            std::vector< std::string > damageSoundNames,
            std::vector< std::string > otherSoundNames )  [inline]
```

Loads sounds.

Parameters

destroySoundNames	
collisionSoundNames	
damageSoundNames	
otherSoundNames	

8.12.3.3 playSound()

```
bool Object::playSound (
            const std::string & name )  [inline]
```

Plays a sound by name.

Parameters

name	name of the sound

Returns

true if sound was played

false if sound was not found

8.12.3.4 TakeDamage()

```
virtual bool Object::TakeDamage (
            float dmg )  [inline], [virtual]
```

Takes damage, plays sound.

Parameters

dmg	damage taken

Returns

true if destroyed

8.12.3.5 transferScore()

```
float Object::transferScore ( )  [inline]
```

Gives score if destroyed, resets.

Returns

score to add

8.12.3.6 updateTexture()

```
virtual void Object::updateTexture (
            float deltaTime )  [pure virtual]
```

Delete object in timer_s seconds (deleted by Level).

Objects need to be deleted between steps

and sounds take time to finish.

Parameters

timer_s	time to deletion

Implemented in ExplodeBird, SpeedBird, NormalBird, Pig, and Block.

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

- src/objects/object.hpp

8.13 ObjectCollisions Class Reference

Inheritance diagram for ObjectCollisions:

Collaboration diagram for ObjectCollisions:

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

- src/objects/object.hpp

8.14 ObjectDefs::ObjectDefaults Struct Reference

Default values for objects.

`#include <object_defs.hpp>`

Public Attributes

- b2BodyDef **bodyDef**
- std::unique_ptr< b2Shape > **shape**
- float **density** =0.0f
- float **maxHp**
- float **spriteWidth**
- float **spriteHeight**
- std::vector< std::pair< std::string, float > > **normalTextures**
- std::vector< std::pair< std::string, float > > **damageTextures**
- std::vector< std::string > **destroySoundNames**
- std::vector< std::string > **collisionSoundNames**
- std::vector< std::string > **damageSoundNames**
- std::vector< std::string > **otherSoundNames**

8.14.1 Detailed Description

Default values for objects.

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

- src/objects/object_defs.hpp

8.15 Pig Class Reference

Pig class.

`#include <pig.hpp>`

Inheritance diagram for Pig:

Collaboration diagram for Pig:

Public Member Functions

- Pig (b2World *world, float x, float y, ObjectDefs::ObjectDefaults *defaults, float rot=0.0f)

  Construct a new Pig object.
- virtual void **checkDamage** () override
- virtual void updateTexture (float deltaTime) override

  Delete object in timer_s seconds (deleted by Level).

Additional Inherited Members

8.15.1 Detailed Description

Pig class.

No subclasses since pigs are functionally the same.

8.15.2 Constructor & Destructor Documentation

8.15.2.1 Pig()

```
Pig::Pig (
            b2World * world,
            float x,
            float y,
            ObjectDefs::ObjectDefaults * defaults,
            float rot = 0.0f )  [inline]
```

Construct a new Pig object.

Parameters

defaults	default values

8.15.3 Member Function Documentation

8.15.3.1 updateTexture()

```
virtual void Pig::updateTexture (
            float deltaTime )  [inline], [override], [virtual]
```

Delete object in timer_s seconds (deleted by Level).

Objects need to be deleted between steps

and sounds take time to finish.

Parameters

timer_s	time to deletion

Implements Object.

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

- src/objects/pig.hpp

8.16 Player Class Reference

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

- src/player.hpp

8.17 RayCastHitFirst Class Reference

Raycasting for explosions.

```
#include <object.hpp>
```

Inheritance diagram for RayCastHitFirst:

Collaboration diagram for RayCastHitFirst:

Public Member Functions

- float **ReportFixture** (b2Fixture *fixture, const b2Vec2 &point, const b2Vec2 &normal, float fraction)

Public Attributes

- b2Fixture * **hitLatest** = nullptr
- b2Vec2 **hitPoint**

8.17.1 Detailed Description

Raycasting for explosions.

Hits first target in the way

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

- src/objects/object.hpp

8.18 Slingshot Class Reference

Launches objects.

`#include <slingshot.hpp>`

Public Member Functions

- **Slingshot** (float x, float y)
- void launchObject (Object *object)

  Shoots object towards slingshot. Impulse depends linearly on distance from slingshot.
- b2Vec2 getLaunchImpulse (Object *object)

  Get the impulse object would be launched with.
- void drag (Object *object, float x, float y)

  Drag slingshot and object to x, y. Meant to be called every frame when dragging.
- void release (Object *object, float x, float y)

  Counterpart of drag. Launches the object.
- float **getRadius** ()
- void **setPos** (float x, float y)
- void **setPos** (b2Vec2 newPos)
- b2Vec2 & **getPos** ()
- sf::Sprite & **getSprite** ()

Protected Member Functions

- void **loadSounds** ()

Protected Attributes

- b2Vec2 **pos**
- b2Vec2 **launchPos**
- float **maxRadius** = 2.5f
- float **powerMult** = 50.0f
- sf::Sprite **sprite**
- std::vector< std::string > **textures** = { "slingshot1", "slingshot2", "slingshot3" }
- std::map< std::string, sf::Sound > **sounds**

8.18.1 Detailed Description

Launches objects.

8.18.2 Member Function Documentation

8.18.2.1 drag()

```
void Slingshot::drag (
            Object * object,
            float x,
            float y )  [inline]
```

Drag slingshot and object to x, y. Meant to be called every frame when dragging.

Parameters

object	object in slingshot
x,y	drag here (limited by maxRadius)

8.18.2.2 getLaunchImpulse()

```
b2Vec2 Slingshot::getLaunchImpulse (
            Object * object )  [inline]
```

Get the impulse object would be launched with.

Parameters

object	to launch

Returns

b2Vec2 impulse

8.18.2.3 launchObject()

```
void Slingshot::launchObject (
            Object * object )  [inline]
```

Shoots object towards slingshot. Impulse depends linearly on distance from slingshot.

Parameters

object	launch this

8.18.2.4 release()

```
void Slingshot::release (
            Object * object,
            float x,
            float y )  [inline]
```

Counterpart of drag. Launches the object.

Parameters

x,y	point of release

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

- src/slingshot.hpp

8.19 SoundManager Class Reference

Manages sound resources in the game.

`#include <sound_manager.hpp>`

Public Member Functions

- **SoundManager** (const SoundManager &)=delete
- SoundManager & **operator=** (const SoundManager &)=delete

Static Public Member Functions

- static bool playMusic (const std::string &name)

  Play music file. Stops current music.
- static void **stopMusic** ()
- static void setMusicVolume (float volume)

  Set Music Volume.
- static float **getMusicVolume** ()
- static void loadSound (const std::string &name, const std::string &filePath)

  Loads sound into memory.
- static const sf::SoundBuffer * getSound (const std::string &name)

  Get sound with name.
- static bool **hasSound** (const std::string &name)
- static void loadAllSounds ()

  Loads all necessary sounds into memory.
- static void **releaseResources** ()

8.19.1 Detailed Description

Manages sound resources in the game.

8.19.2 Member Function Documentation

8.19.2.1 getSound()

```
static const sf::SoundBuffer* SoundManager::getSound (
            const std::string & name )  [inline], [static]
```

Get sound with name.

Parameters

name	saved name

Returns

const sf::SoundBuffer* sound

8.19.2.2 loadSound()

```
static void SoundManager::loadSound (
            const std::string & name,
            const std::string & filePath )  [inline], [static]
```

Loads sound into memory.

Parameters

name	saved name
filePath	sound file path

8.19.2.3 playMusic()

```
static bool SoundManager::playMusic (
            const std::string & name )  [inline], [static]
```

Play music file. Stops current music.

Parameters

name	Name of music file. See initialization.

Returns

true if success

8.19.2.4 setMusicVolume()

```
static void SoundManager::setMusicVolume (
            float volume )  [inline], [static]
```

Set Music Volume.

Parameters

volume	volume 0-100

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

- src/sound_manager.hpp

8.20 SpeedBird Class Reference

Inheritance diagram for SpeedBird:

Collaboration diagram for SpeedBird:

Public Member Functions

- **SpeedBird** (b2World *world, float x, float y)
- void Attack () override

  Bird does a special attack.
- virtual void updateTexture (float deltaTime) override

  Delete object in timer_s seconds (deleted by Level).
- virtual void **setDestroyTexture** () override

Protected Attributes

- const float **abilitySpeedGain** = 20.0f
- std::vector< std::vector< size_t > > **texture_order** = {{0,1}, {0,2}, {0,3}}
- size_t **row** = 0
- size_t **column** = 0

Additional Inherited Members

8.20.1 Member Function Documentation

8.20.1.1 updateTexture()

```
virtual void SpeedBird::updateTexture (
            float deltaTime )  [inline], [override], [virtual]
```

Delete object in timer_s seconds (deleted by Level).

Objects need to be deleted between steps

and sounds take time to finish.

Parameters

timer_s	time to deletion

Implements Object.

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

- src/objects/special_birds.hpp

8.21 TextureManager Class Reference

Manages textures in the game.

```
#include <texture_manager.hpp>
```

Public Member Functions

- **TextureManager** (const TextureManager &)=delete
- TextureManager & **operator=** (const TextureManager &)=delete

Static Public Member Functions

- static void loadTexture (const std::string &name, const std::string &filePath)

  Loads texture into memory.
- static sf::Texture & getTexture (const std::string &name)

  Gets a texture based on the saved name.
- static bool **hasTexture** (const std::string &name)
- static void loadAllTextures ()

  Loads all necessary textures into memory.

8.21.1 Detailed Description

Manages textures in the game.

8.21.2 Member Function Documentation

8.21.2.1 getTexture()

```
static sf::Texture& TextureManager::getTexture (
            const std::string & name )  [inline], [static]
```

Gets a texture based on the saved name.

Parameters

name	Name of the texture

Returns

const sf::Texture& the texture

8.21.2.2 loadTexture()

```
static void TextureManager::loadTexture (
            const std::string & name,
            const std::string & filePath )  [inline], [static]
```

Loads texture into memory.

Parameters

name	Name the texture will be saved with
filePath	path to texture file

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

- src/visual/texture_manager.hpp

Chapter 9

File Documentation

9.1 tests/tests.cpp File Reference

Using tests:

```
#include <iostream>
#include <gtest/gtest.h>
#include <box2d/box2d.h>
#include "../src/objects/block.hpp"
#include "../src/objects/object_defs.hpp"
#include "../src/objects/special_birds.hpp"
#include "../src/level.hpp"
```

Include dependency graph for tests.cpp:

Functions

- **TEST** (BlocksTests, Create)
- **TEST** (BlockTests, TakeDamage)
- **TEST** (BlockTests, Destroy)
- **TEST** (BirdTests, Create)
- **TEST** (BirdTests, Attack)
- **TEST** (LevelTests, Load)

9.1.1 Detailed Description

Using tests:

1. Build project
2. Run 'ctest' in /build

Alternatively use the 'Testing' tab in VS Code