Cut The Rope

The Goal : Get the candy back to Iggy, using variety of elements.

==============================================================================

Written by member 1 : Yehonatan Bakshi id : 308077668 login : yehonatanba

Written by member 2 : Hila Saadon id : 208405217 login : hilasaa

Description :

The gameplay is based on delivering the candy to Iggy who is trying to save the doomed galaxy.

Iggy was sent to an intergalactic mission to reach to the divined Candy.

The game is game levels genred as Puzzle Logic. The player can effect the movement of the Candy using various of equipments, cutting ropes, all under the gravity forces. (Secret : some elements can change gravity forces).   
Each stage contains stars to collect as grade and different elements of equipments.

Listed :

Rope, Blower, Bubble, Hat Teleport, Gravity Changer, Spike.

Don’t forget to look at the clock, some levels are on limited time.

collect stars and use various equipment to move the candy.

The gameplay is based on delivering the candy to Iggy by cutting the ropes

Input file format :

First line is the limited time -1 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

In the resource directory there will be a file named board.txt that will be

holding all the level file's names.

Currently the file holds the names Level1.txt, Level2.txt, Level3.txt, Level4.txt.

Each “LevelX.txt” file hold the build of a map and their order in the file points at the order that they will be loaded and presented to the user

(for this scenario it’s a numerical order 1-2-3-4 for the 1st, 2nd, 3rd and 4th levels).

Each LevelX.txt is being scanned as each line represents an object by 4 parameters :

ID(Enum) Extra info(by need) location.x location.y

Note : “Extra info” is being used to read needed data for some objects such as :  
– angle for Blower, Hat Port.  
– quantity of joints to create rope.  
If not being used = 0 .

Const numbered objects (ID to recognize objects while scan file of level):  
0 = Frog  
1 = Candy  
2 = Limit  
3 = Rope  
4 = Rope Base  
5 = Star  
6 = Spike  
7 = Teleport  
8 = Blower  
9 = Bubble  
10 = Gravity Set

**Full GameObject(s) List:**

Physics Object : (has a body, under gravity forces)  
**1.** Candy – The main gameplay element, under gravity force.   
**2.** Rope – Attached to the candy, click and pull to cut.

Non-Physics Object :  
**1.** Blower – Static. Blows air and effect the Candy if it’s in the zone. Placed by angle.   
**2.** Bubble – Static (While not on candy). When Candy touches the bubble, it will integrated and change the gravity of the candy to fly upwards. Can be popped with a single tap, then the candy’s gravity will be switched back to normal.   
**3.** Frog – Static. Our beloved Iggy which symbolize the wanted destination for candy.  
**4.** Gravity Change – Static, Button to change gravity of the entire world  
**5.** Hat Teleport – Static, two ports that the Candy can transform from one to another. (Both are entry and exit).  
**6.** Limit – Static, Global bounds of the world which are symbolized by pink clouds.  
**7.** MouseTool – click and pull to manage and create a cut, check if intersection with rope.   
**8.** RopeBase – Symbolize starting point of the rope, anchor.  
**9.** Spike – Static, crash candy if collide.  
**10.** Star – Static, 3 stars is the max grade on every level, can be collected by Candy.

**Files list :**

**Include (header) section files .h :**

**1.** Animation.h :  
Handling and apply the animation.

**2.** AnimationData.h :  
Holds the information for animation.

**3.** Blower.h :  
Holds the class type, includes and relevant functions such as set and get.  
Contain a transparent rectangle as the blow zone.

**4.** Bubble.h :  
Holds the class type, includes and relevant functions such as set and get.

**5.** Button.h :  
Holds the class type and includes, set position of buttons and allows to start handling pressing them.

**6.** Candy.h :  
Holds the class type, includes and relevant functions such as set and get.  
Contains functions about candy’s condition – if bubbled, current gravity, number of collected stars…

**7.** Collision.h :  
Taken from SFML class to handle collision PixelPerfectTest.

**8.** CollisionHandling.h :  
Struct organized of Multi Method to handle collisions between objects.

**9.** Controller.h :  
Holds the class type and includes, allows to run game and menu, draws needed information for them and updates the game objects.

**10.** DataToolBar.h :  
Manage the technical information of tool bar, such as : Level, Time since game started, Star score and limited time (if limited).

**11.** Frog.h : Holds the class type, includes and relevant functions such as set and get.  
The destination of the candy.

**12.** GameObject.h :  
Holds the class type and includes, holds a type for each game object, set and get position of object, sprite…

**13.** GravityChanger.h :  
Holds the class type, includes and relevant functions such as set and get.

**14.** HatPort.h :  
Holds the class type, includes and relevant functions such as set and get, Pairing teleports.

**15.** LevelManager.h :  
Holds the class type and includes, holds information and functions regarding the level that is being read from .txt files such as time…

**16.** Limit.h :  
Holds the class type, includes and relevant functions such as set and get.

**17.** Macros.h :  
Holds all of the Const values we need to use in the program.

**18.** Menu.h :  
Holds the class type and includes, holds menu's draw function and handling mouse events (mouse pressing buttons).

**19.** MouseTool.h :   
Holds the class type, includes and relevant functions such as set, get, draw.  
Check if the click and pull is big enough to be a cut, , check if intersection with rope.

**20.** NonPhysicsObject.h :  
Holds the class type and includes, allows to inherent any functionality to any non-physics objects.

**21.** PhysicsObject.h :  
Holds the class type and includes, allows to inherent any functionality to any physics objects.  
create body as requested, according to the object.

**22.** Resources.h :  
Using Singletone. Holds the class type and includes, hold all resources such as enum of sounds, fonts, textures and animation to recognize each resource file.

**23.** Rope.h :  
Holds the class type, includes and relevant functions such as set and get.  
Receive head and tail bodies, create joints and connect them to create a rope, define the properties of body.

**24.** RopeBase.h :  
Holds the class type, includes and relevant functions such as set and get.

**25.** Spike.h :  
Holds the class type, includes and relevant functions such as set and get.

**26.** Star.h :  
Holds the class type, includes and relevant functions such as set and get.

**27.** World.h :  
Holds the class type and includes, controls the world, level functions and updates the objects, data tool, sounds and all the occurrences ingame, allows to read data from mouse click and so.

**Source Section files .cpp :**

**1.** Animation.cpp :  
Handling and apply the animation.

**2.** AnimationData.cpp :   
None.

**3.** Blower.cpp :  
Implementation needed functions for the Blower object.

**4.** Bubble.cpp :  
Implementation needed functions for the Bubble object.

**5.** Button.cpp :  
Holds the class type and includes, controls the location of buttons and allows to start handling pressing them.

**6.** Candy. cpp :  
Implementation needed functions for the Candy object.  
Functions set and get information about candy, such as : if bubbled, lost, collected stars and so…

**7.** Collision. cpp :  
Taken from SFML class to handle collision PixelPerfectTest.

**8.** CollisionHandling.cpp :  
Handling all the collisions in game between objects. Mostly it will be between Candy and any other game object.

**9.** Controller.cpp :  
Holds the class type and includes, allows to run game and menu, draws needed information for them and updates the game objects.

**10.** DataToolBar.cpp :  
Implementation needed functions for data tool bar for time, time left, level number and star score of current level.

**11.** Frog.cpp :  
Implementation needed functions for the Frog object.

**12.** GameObject.cpp :  
Holds the class type and includes, holds a type for each game object there is (such as king fire and etc...) and handles their collision.

**13.** GravityChanger.cpp :  
Implementation needed functions for the Gravity Changer object.

**14.** HatPort.cpp :  
Implementation needed functions for the Hat Teleport object, pairing teleport.

**15.** LevelManager.cpp :  
Holds the class type and includes, holds information and functions regarding the level that is being read from .txt files such as time, level design etc...

**16.** Limit.cpp :  
Implementation needed functions for the Limit object.

**17.** Menu.cpp :  
Holds the class type and includes, holds menu's draw function and handling mouse events (mouse pressing buttons).

**18.** MouseTool.cpp :  
Holds the class type, includes and relevant functions such as set, get, draw.  
Check if the click and pull is big enough to be a cut, , check if intersection with rope.

**19.** NonPhysicsObject.cpp :  
Holds the class type and includes, allows to inherent any functionality to any non-physics objects.

**20.** PhysicsObject.cpp :  
Holds the class type and includes, allows to inherent any functionality to any physics objects.  
create body as requested, according to the object.

**21.** Resources.cpp :  
Using Singletone. Holds the class type and includes, hold all resources such as enum of sounds, fonts, textures and animation to recognize each resource file and loads it.

**22.** Rope.cpp :  
Implementation needed functions for the Rope object.  
Receive head and tail bodies, create joints and connect them to create a rope, define the properties of body

**23.** RopeBase.cpp :  
Implementation needed functions for the Rope Base object.

**24.** Spike.cpp :  
Implementation needed functions for the spike object.

**25.** Star.cpp :  
Implementation needed functions for the star object.

**26.** World.cpp :  
Holds the class type and includes, controls the world, level functions and updates the objects, data tool, sounds and all the occurrences ingame, allows to read data from mouse click and so.  
Holds vector of NonPhysicsObject.

**Files :**

**Audio files :**  
**1.** 50Cent - Candy Shop(Instrumental Remix)(Game Music).wav   
**2.** AlakazamTeleport.wav  
**3.** BlowingEffect.wav  
**4.** Bubble2CandyEffect.wav  
**5.** BubblePopEffect.wav  
**6.** ButtonClick.wav  
**7.** ByeBye.wav  
**8.** CutRopeEffect.wav  
**9.** DoorKnock.wav  
**10.** DoorOpen.wav  
**11.** GravityEffect.wav  
**12.** LoseCartoon.wav  
**13.** OhNoEffect.wav  
**14.** StarCollectedEffect.wav  
**15.** The Chordettes - Lollipop(Menu Music).wav  
**16.** WaitingYoohooEffect.wav  
**17.** WinYummyEffect.wav  
  
**Texuture files :  
1.** AnimationBlower.png  
**2.** AnimationCandy.png  
**3.** AnimationFrog.png  
**4.** AnimationStar.png  
**5.** AnimationStarScore.png  
**6.** Back.png  
**7.** Bubble.png  
**8.** Exit.png  
**9.** GameBackground.png  
**10.** GravityChanger.png  
**11.** HatTeleport.png  
**12.** Help.png  
**13.** HelpData.png  
**14.** LevelUpMsg.png  
**15.** Limit.png  
**16.** LoseMsg.png  
**17.** MenuBackground.png  
**18.** Reset.png  
**19.** Rope.png  
**20.** RopeBase.png  
**21.** Spike.png  
**22.** Start.png  
**23.** StartMsg.png  
**24.** WinMsg.png

**Animation-Texts files :  
1.** Blower.txt  
**2.** Candy.txt  
**3.** Frog.txt  
**4.** GameBackground.txt  
**5.** Star.txt  
**6.** StarScore.txt  
  
**Board Files :  
1.** Board.txt - hold level's names (map designs). **2.** Level1.txt - holds first level design **3.** Level2.txt - holds second level design **4.** Level3.txt - holds third level design **5.** Level4.txt - holds fourth level design

**Main data bases :**

Multi-Method

Struct to hold animation data

Struct to hold LevelData details such as : Level number, Time, Time Left.