

technical support manual

SulAxya lama

kAYLA harrison

Ayça Küçükdurmaz

Space Game

CS 240 Team

2016 Edition

Table of Contents

[Game Loop Functions 2](#_Toc400051163)

Game Object Functions……………………………………………………………………….......9

SDL Functions...............................................................................................................................15

Game Loop Function

[Game Loop 3](#_Toc400051158)

[Score 4](#_Toc400051159)

[Checkbounds 5](#_Toc400051160)

Run……………………………………………………..................................................………….6

Game Objects.........………………………………………………………………………………..7

Bullet Reallocation………………………………………………………………………………...8

Game Loop

**Prototype:**

Gameloop( );

**Parameters:**

None

**Description:**

Loop for the game and reset the score to 0.

Score

**Prototype:**

int getScore();

void setScore(int );

**Parameters:**

int

**Description:**

Setter and Getter for the score.

Checkbounds

**Prototype:**

void CheckBounds (GameObject\* object);

**Parameters:**

GameObject\* object

**Description:**

This function is used for checking bounds to prevent player object from going off screen.

Run

**Prototype:**

void run();

**Parameters:**

None

**Description:**

Run function executes the game. It consist of lot of SDL functions including initiating the screen to placing the object on the screen, movement controls of the game. SDL functions used are described in following pages.

Game Objects

**Prototype:**

vector<GameObject\*> gameObjects

**Parameters:**

None

**Description:**

A vector is used to store all the object images for the game. The image are in bitmap format.

Bullet Reallocation

**Prototype:**

void BulletRealloc(GameObject\* object);

**Parameters:**

GameObject\* object

**Description:**

Movement of the bullet. If out of bounds, they will be store in one spot background.

Game Object Function

Load Image 10

Move 11

Getters……………………………………………………………………………………………12

Setters…………………………………………………………………………………………….13

Game Screen…………………………………………………………………………………..…14

Load Image

**Prototype:**

GameObject(std::string imageFilename)

**Parameters:**

string imagefilename

**Description:**

The function is used for loading all the bmp object images and check conditions

Move

**Prototype:**

Move(int newX, int newY)

**Parameters:**

int newX;

int newY;

**Description:**

The function acts as setter for both x and y. It handles moving objects.

Getters

**Prototype:**

virtual int getX();

virtual int getY();

virtual bool getisAbleToShoot();

virtual bool getisShooting();

virtual bool getisAlive();

virtual bool getcollapsed();

virtual bool getisBullet();

virtual bool getisEnemy();

virtual bool getisPlayer();

virtual int getValue();

**Parameters:**

None

**Description:**

The above functions are coded as the getters. Virtual keyword is used. A virtual member is a member function that can be redefined in a derived class, while preserving its calling properties through references.

Setters

**Prototype:**

virtual void setisAbleToShoot(bool);

virtual void setisShooting(bool);

virtual void setisAlive(bool);

virtual void setcollapsed(bool);

virtual void setisBullet(bool);

virtual void setisEnemy(bool);

virtual void setValue(int);

**Parameters:**

bool

int

**Description:**

The above functions are coded as the setters and have parameters and returns either Boolean condition or integers. Virtual keyword is used. A virtual member is a member function that can be redefined in a derived class, while preserving its calling properties through references.

Game Screen

**Prototype:**

void updateScreen(); // mainfunction cycle will have everything below

void loadImages();

void cleanOuttaScreen(); //loop thru gameObject x & y's

void gameDisplayBuffer(); //will decide new object have them ready

void collisionCleaning();

void updateScore();

**Parameters:**

None

**Description:**

This Game Screen class will handle everything that it will be displayed on screen and it'll destruct every game object we don’t see, clean after collisions, create new object on the screen and update score.

SDL Functions

The program was based on SDL. Following code are implemeneted on SDL.

SDL\_Window\* window = 0;

* To create a window

SDL\_Surface\* screenSurface = 0;

* To create a surface to apply images

SDL\_Init(SDL\_INIT\_EVERYTHING); //

* initialize SDL

SDL\_Event e;

* Get and handle SDL events (input, errors, etc)

switch( e.key.keysym.sym )

* Keyboard events

SDL\_QUIT

* to close the window

GameObject\* currentObj = gameObjects.at(i);

* draw a rectangle around the image.

SDL\_Delay(1000.0 / (float)fps)

* for delaying

SDL\_Surface\* ImageLoader::loadImage(std::string filename)

* takes filename as a parameter and load image onto the screen