University of Dhaka

Department of Computer Science and Engineering

Project Report:
Fundamentals of Programming
Lab (CSE-1211)

Project Name:
Falcon-6T : Knights

Team Members:

Afser Adil Olin (47)
Md Shah Nawaz Parvez (21)
Galib Mahmud Jim (62)

1. Introduction

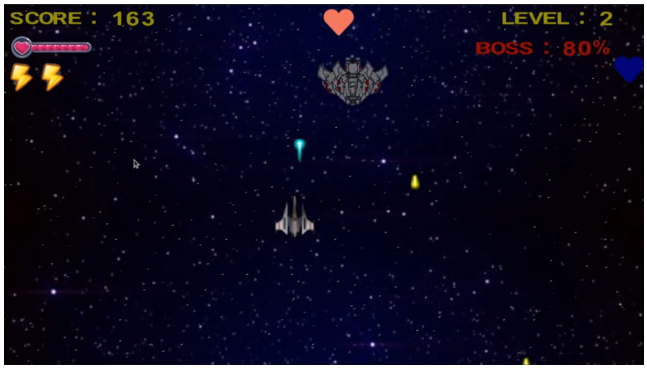
A team of lightly armed and highly trained space soldiers are patrolling around space in their spaceship to look out for possible alien threats. Suddenly they notice that a huge army of heavily armed aliens are incoming toward them. Finding themselves in such a critical situation, the brave space soldiers take on against the whole army with their only lightly armed patrolling spaceship to save their beloved home, Earth. In this game, we will let the users to live the thrill of a real space knight and save the galaxy using various features and battle tactics available in this game.

2. Objectives

- Making a simple 2D game using SDL2 library
- Applying only C/C++ programming language to complete the project
- Exploring the inter-connection between various libraries and C/C++ programming language.

3. Project Features

- 25 levels, Boss battles and Boss health bar



- Primary weapon and Secondary special weapon

- **Life pod, weapon charge pod and health healing system with Player health bar**
    - **Secondary weapon charging/powerup system**

- **Visual Destroying effects following fundamental rules of physics**

- **Player score and Highscore viewing feature**

- Sprinting in movement of player to dodge enemy bullets
- In-game pause/continue feature

- In-game sound effects and intro music
- UI cursor selecting and clicking sound effect
- Dynamic and user friendly UI

4. Project Modules

Main.cpp

memset( ) : sets the memory of all App elements to zero.
initSDL( ) : Initiates SDL Image and Audio, creates Window and shows Cursor.
initGame( ) : calls functions that initiate background, starfield, Instructions , credits,fonts and highscore table.
init_intro( ): loads various image and audio media assets of this game.
intro( ): Prepares scene then draws images that were loaded in init_intro, delays Change in images by mentioned amount of milliseconds and calls MainIntro( ) Function to take us to UI section.

doInput ( ): At first sets all the Input events to zero and then takes both mouse and keyboard input from the user during gameplay.
app.delegate.logic( ): Follows ordered logics during gameplay.
app.delegate.draw( ): Draws essential images to visualise gameplay elements.

Credit.hpp

creditlogic( ) : ESC key for return to the home page and play a sound effect.
creditdraw ( ) : Draw the credit image as crdt SDL_Texture.
init_credit( ) : Load the credit image into the crdt SDL_Texture.
Credit( ) : Call the creditlogic and creditdraw functions into the app.delegate.logic and app. Delegate.draw. Then set the memory of keyboard and mouse to Zero.

draw.hpp

prepareScene( ) : Set the app.renderer color and clear the renderer.
presentScene ( ) : Present every loaded and copied texture and elements into the app.renderer and show it on the window.
addTextureToCache( ) : Add new texture to the memory.
getTexture( ) : Check if texture already exists on memory, else return NULL to show that there is no such texture.
loadTexture( ) : It returns SDL_Texture type and the parameter of this function is an Image file name. Load the file into a Surface . SetColorKey of that surface to make the background of the image transparent then convert it into a Texture and Return it.
blit( ) : Copy a texture into the app.renderer . It has three parameters. First one is that texture and two integers x and y, which indicates the position of the window where the picture will blit.
blitRect(SDL_Texture *texture, SDL_Rect *src, int x, int y) : Copy the texture into the app.renderer in the position of x and y of the window. Src indicates the position in the source texture, height and width we want to copy into the render.
blitFont(SDL_Texture *texture, SDL_Rect *src, int x, int y, int w, int h) : It blit the bitmap font x and y position in the window with w width and h height.

Intro.hpp

init_intro( ): loads various image and audio media files
select(x,y,z): creates a rectangular box which will appear in Main Menu when cursor is taken on a particular mentioned region.

homelogic( ): when the cursor is clicked on each option in the main menu, this module takes the user to that particular feature. When clicked on New game, starts a new game by calling initstage( ). when clicked on highscore , shows the scoreboard by calling initHighscores( ) and so on. Also plays a click sound when the mouse is clicked .

Homedraw( ): draws a green rectangle on the available options in the main menu.

homepage( ): calls homelogic and homedraw and sets all the keyboard and mouse events to zero when first entered in the main menu.

ilogic( ): Do all logical tasks for the intro page.

idraw( ): Draw the intro page.

MainIntro( ): Call the main intro page.

intro( ): Begins intro when the game is first started. draws various intro images and then calls mainIntro to take user to the Main Menu.

instruction.hpp

instlogic( ): When clicked on Play, inside the instruction page, starts the game by calling initstage( ). And when clicked on Main Menu, Takes the user back to the Main Menu page by calling homepage( ). Also plays click sound when pressed on play or main menu.

instdraw( ): draws a green rectangle around the Play or Main Menu button when the cursor is taken upon any of them.

init_instruction( ) : Loads the image file to show inside the instruction.

Instruction( ): calls instlogic and instdraw to enable menu clicking and selecting features. also sets all the keyboard and mouse events to zero .

init.hpp

initSDL( ): Creates window, renderer and shows cursor on the window.

initGame( ): initiates game background by calling initBackground() function. Initiates stars on display by calling initStarfield. loads instruction texture and credit texture when init_instruction and init_credit functions are called. Initiates fonts for showing highscore, score, boss health percentages by calling initfont() function. Initiates Highscore table features by calling initHighscoreTable() function.

stage.hpp :

initStarfield( ): This function loads the star field that we saw in the background.
initBackground( ) : Load the background
initPlayer( ) : Load all data for the player.
doBackground( ) : Move the background.
doStarfield( ) : Move the star field.
capFrameRate( ) : To calculate the precise 60 frame per second (instead of SDL_Delay( ))
addDebris( ) : Creating visual effects for destroyed Entities
spawnUlt( ) : Spawn ultimate move for the player
doultEnergy( ) : Move the fire effect that created on spawnUlt
addExplosions( ) : To add destruction effects for the ships that have been destroyed.
doExplosions( ) : Change its position and check if the time limit is over, then delete this from memory.
doDebris( ) : Move the broken part or debris in the screen. If the debris is outside of the screen then delete those from memory.
bulletHitfight( ) : Check if any bullet from the opposite side hits any fighter i.e. enemy of player.
doBullet( ) : If any bullet hits any fighter or get outside of the screen then delete the bullet. It also changes the position of the bullets according to its path.
fireAlienBullet( ) : If the reload time for an enemy get 0 or less, it fire a bullet by this function.
doEnemy( ) : Check all enemies, if their reload time gets zero then fire a bullet from it and reset its reload time.
doPod( ) : Check the pods, if it collides with the player then give some buff according to its power. It also changes its position according to its path and checks if it is out of the screen or collides with the player. Then just remove it from memory.
AddPod( ) : If an enemy is destroyed then create a pod randomly by this function.
doFightere( ) : Changes the position of all enemies and players according to their path. If anyone died then make an explosion and add them to debris and delete the Fighter from the memory.
addBossBullets( ) : Fire the ultimate move from the boss .
doBoss( ) : Change the position of the boss. If the boss's life gets 0 then tell us that the boss is dead.
spawnBoss( ) : Spawn a boss in every level after 1 min.
resetStage( ) : reset everything before the game starts.
logic( ) : All logical calculations done here.
initstage() : The game starts from here.

defs.hpp :

Define some important number as a name, so that it is easy to read the code.

Highscore.hpp :

drawHighscores( ) : Draw the highscores that are saved .

drawInputName ( ) : Draw while taking the name input for the score.

doNameInput() : Take the name as input.

highscoreComparator( ) : Compare between to score.

addHighscore( ) : Add a score to the high score table after the game is finished.

hlogic ( ) : Logic function for the highscore part.

hdraw ( ) : Draw the Highscores and other stuff in that page.

initHighscores ( ) : hlogic and hdraw function is called from here.

initHighscoreTable ( ) : Initialize highscores from the saved file HighScores.txt at the start of the game.

text.hpp

initfont( ) : Load the Bitmap font image and set GLYPH_HEIGHT and GLYPH_WIDTH value.

movement.hpp :

spawnenemy( ) : Spawn the enemy ship. Increase the enemy ship spawn time with the level

fireBullet( ): Fire the bullets at the enemy ship.

movePlayer( ) : move the player Left, Right, Up, Down. Shift key for stop firing and speed up the movements.

Util.hpp :

collision( ) : Return yes when two objects collide with each other. Example: enemy bullets and player ship, Player bullets with enemy ship.

calcSlope( ) : Calculate the slope between two entities from their position. It helps us to find the path for enemy bullets towards the player.

Input.hpp( ) : Getting all kinds of input from this header file.

structure.hpp( ) : Every important variables and structure that are requires are implemented here.

5. Team Member Responsibilities

Afser Adil Olin :

- *Stage*
- *Level*
- *In game logical interfaces (boss,player,level)*
- *UI*
- *Input*
- *Draw*
- *collision*

Md Shah Nawaz Parvez :

- *Graphics*
- *Intro*
- *UI*
- *Sound Effect*
- *mouse events*
- *credit*
- *Instructions*
- *Level effects*

Galib Mahmud Jim :

- *Bitmap fonts*
- *Highscore*
- *UI*
- *Image and font Manipulation*
- *Intro*
- *Alpha Blending*
- *Sound Effects*

6. Platform, Library & Tools

- **Platform : C++, Makefile**
- **Library : SDL-2**
- **Tools: Visual Studio Code, GitHub, GNU Image Manipulation Program**

7. Limitations

We failed to render video/gif format files for the intro. Also, due to inexperience in graphics design we failed to come up with more interesting and challenging features.

8. Conclusions

The logics implemented in this game are well thought and implimented. The code is divided into little modules which makes it easier to add , remove or change various features without facing much hassles. The variables are well documented, making it easier for future development and understanding. The intro, graphical rendering and the UI are simple and user friendly. The gameplay is smooth and flawless. Rules of physics are correctly followed as much as possible. The gameplay is very interesting and becomes more interesting as the player levels up one by one. The destruction, shooting and movement during the gameplay is also bug free. The implemented special features, such as halt of shooting, shifting while moving, also leave room for creativity of the player.

9. Future plan

In future, We want to render gif and video files to give the user more premium feeling. We plan to implement both keyboard and mouse inputs simultaneously. We plan to introduce more thrilling in-game futures to make the gameplay more challenging and thrilling. We plan to add many new levels and plot-arena which will keep challenging the player's ability to level up.

Repositories

GitHub Repository: *https://github.com/adil-olin/Project-Falcon-6T-Knight*

Youtube Video: *https://youtu.be/oVtPAjs8wBA*

References

Sites used for support:

- https://lazyfoo.net/tutorials/SDL/
- https://wiki.libsdl.org/APIByCategory
- https://stackoverflow.com/
- https://github.com/
- https://www.logodesign.net/gaming-logo-maker