University of Dhaka

Department of Computer Science and Engineering

Project Name :

ProjectZ

Team Members

1. Reyadath Ullah (Roll : 33)
2. Ittehad Saleh Chowdhury (Roll : 11)

1. Introduction

To summarize, our project is a collection of retro games. Currently it consists of 4 different games. They are :

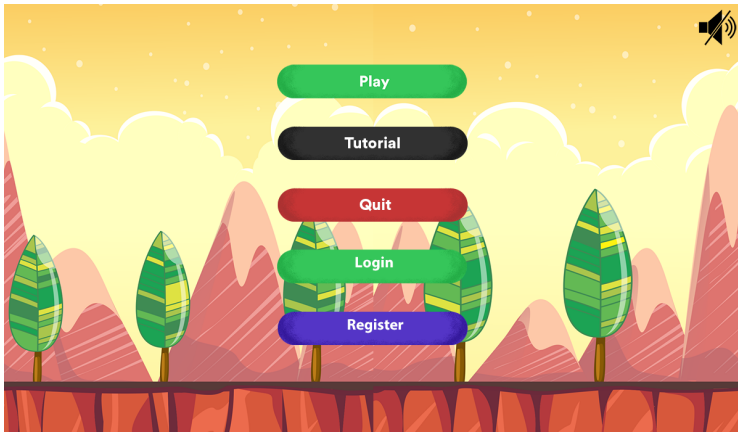
- Bucket Ball
- Pacman
- Icy Tower
- Space Runner

These are variants of some really well known classics. One of the interesting parts of the game is that players can navigate through the games using an open world RPG system. A character can freely move around the map and within that map, these games will be spawned at certain positions. By moving your character to those positions you can start a game. Each game has it's separate highscore board and consist of it's own menu options.

2. Objectives

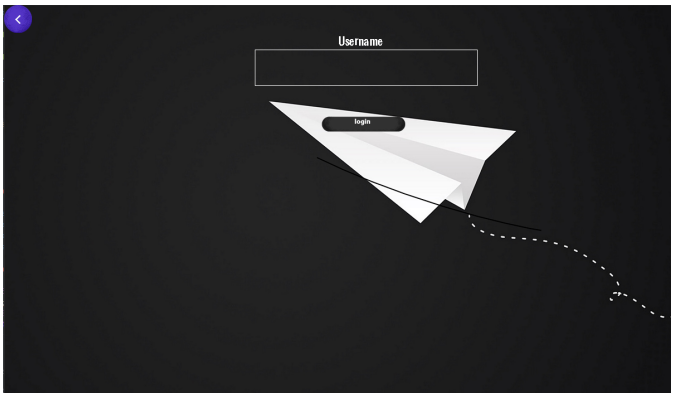
The main objective of our project is to make use of everything we have learned on SDL and make a fun Game enjoyable for everyone.We decided to revive our childhood memories by making a simple open world RPG game that consists of several classic games that we have played in our childhood and to implement it in a way that makes future extension very easy and simple.

3. Project Features



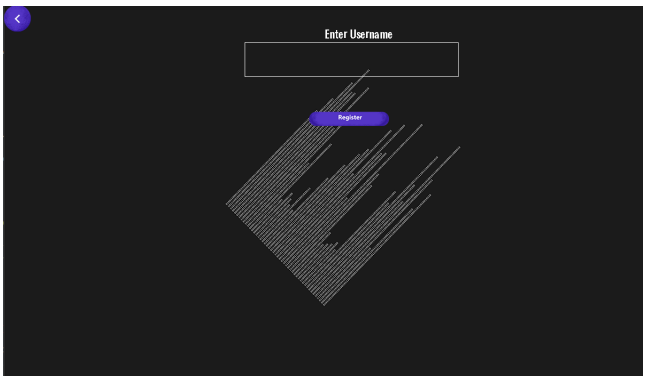
3.1 Main Menu

The main menu has all the basic functionality that you'd expect from a game such as "Play" and "Quit" button. Clicking the play button would start the game and pressing quit will exit the game. The interesting feature here is the "Login" and "Register" Menu.



3.1.1 Login Menu

Clicking the login button will take you to the page shown on the right. If you're already registered you can enter the username and click login. Basic text input functionalities such as copy paste have been implemented as well. Clicking the ESCAPE key or by clicking the back button on the top left corner will take the user back to the main menu. This will take you to the main game logged in as the given username. The purpose for this feature is to avoid adding your username to save score multiple times as this project contains multiple games with separate high score boards.

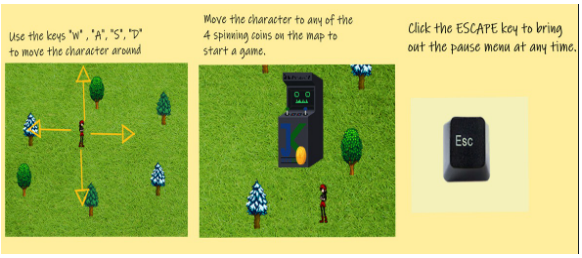


3.1.2 Register Menu

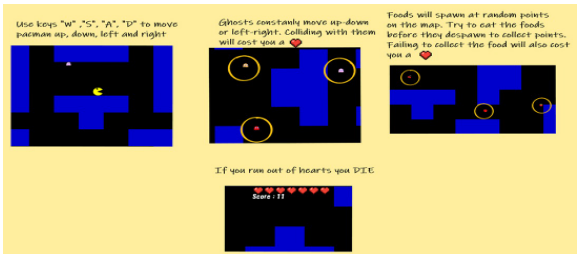
Clicking the Register button on the menu will bring the user to the page shown on the left. You can register simply by entering a username that's not already taken and clicking the register button. Similar to the login menu, the user can go back by clicking the escape key or the blue back button.

3.1.3 Tutorial Menu

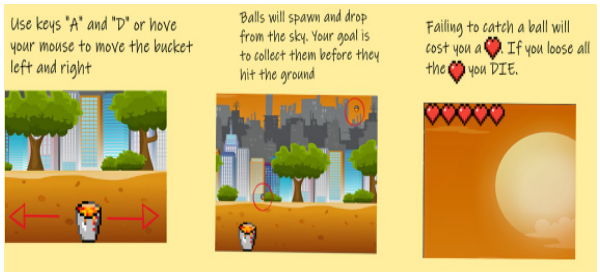
The main menu and the menu in each of the 4 games contain a Tutorial button.Clicking the tutorial button will give you a brief description of the controls and the main objective of the games.



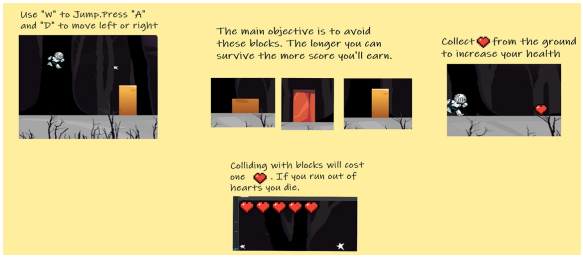
Main menu Tutorial



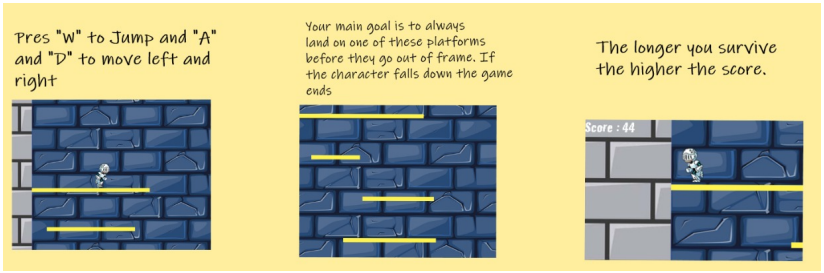
Pacman Tutorial



Bucket Ball Tutorial



Space Runner Tutorial



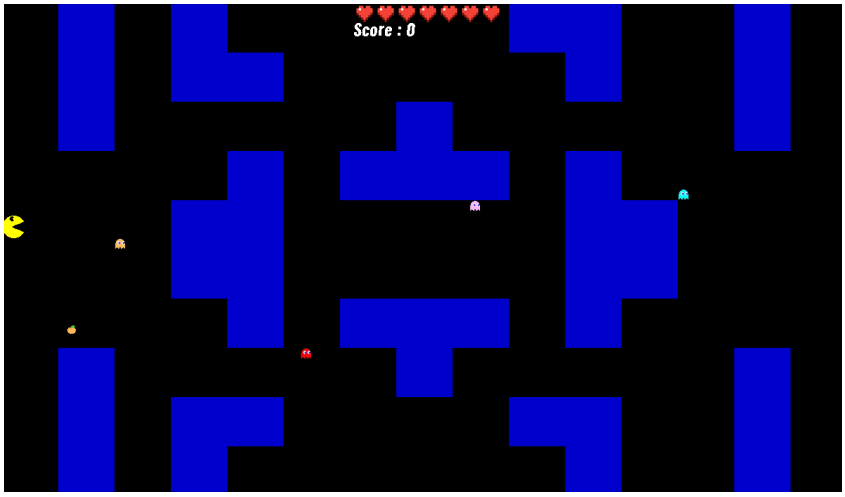
Icy Tower Tutorial

3.1.4 Volume Control

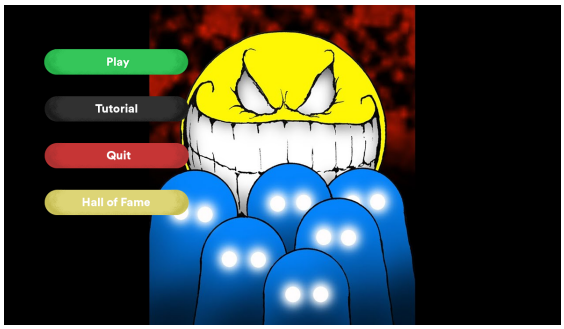
Clicking the speaker icon on the top right corner of the screen on the menu options will turn on or off the background music. If the music is on it'll show the icon on the first image otherwise the icon on the second image.

3.2 Pacman Game

We tried to recreate a simpler version of the classic Pacman game.



3.3.1 Pacman Menu:



The Game starts with a menu having some basic features like the “Play” “Quit” button.The “Tutorial” button gives you a brief description of how to play the game and few rules.”The Hall of Fame” button takes you to a list that contains the top 10 high scores achieved by different players until now

3.2.2 Randomly spawned food:

There are 5 types of food and they spawn randomly at different parts of the map.After spawning each food stays for a specific amount of time and the player must eat them before they disappear.Each type of food rewards a specific amount of points.Failing to eat a food before disappearing will cost the player a life displayed on the upper part of the window.

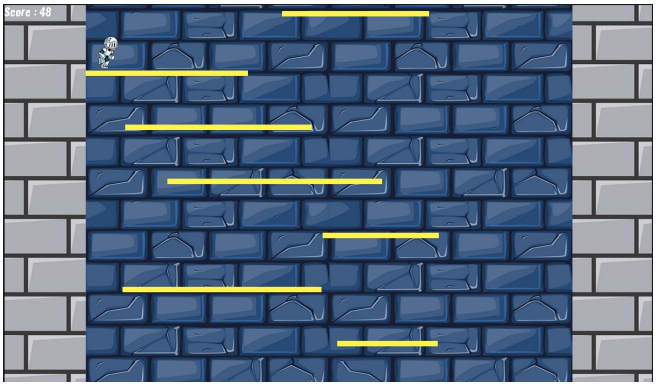
3.2.3 Moving Ghosts:

There are 4 types of ghosts and they move horizontally or vertically within a specific range.If the pac man touches any of the ghosts the player will lose a life and will be respawned at the centre of the map as penalty.

3.2.4 Automatic scoring system and collision detection:

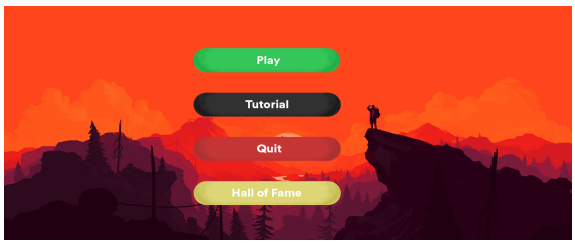
The Pacman will stop as soon as it collides with any of the obstacles.Also if the pacman collides with any of the food the score will increase and if it collides with any of the ghosts or fails to eat any food before it disappears the lives at the top will decrease simultaneously.

3.3 Icy Tower



Icy Tower is a classic game where you have to jump from one platform to another and try not to fall down.As you progress further and further into the game the speed of the platforms increases making it harder to survive.The score increases as long as you are able to survive in the game.

3.3.1 Icy Tower Menu:



The game starts with a menu containing general “Play” “Quit” and a “Tutorial” button that gives a brief description of how to play the game.”Hall of fame” button contains a list of top 10 high scores.

3.3.2 Moving platforms:

The platforms spawn randomly although a check function is used to make sure such that the platforms are not so far apart so that it becomes impossible for the character to jump to the next platform.Also all the platforms move at a specific velocity downward.New platforms are added as soon as one of them disappear under the window.We also scroll the background to create the illusion of the character moving up.

3.3.3 Running Animation:

A running animation of the character happens while running left or right.

3.3.4 Jumping mechanics:

The jumping mechanics are implemented in such a way such that the character goes through the platforms while moving from up but stops and lands on them while coming from down.Also the character can move left and right while jumping and can cover horizontal distance on air.

3.4 Bucket Ball Game

This is a simple game where balls fall down from the sky randomly and the objective is to catch all the balls in the bucket.Dropping a ball will cost the player a life shown on the upper left side of the window.Catching each ball reward a specific amount of points that is added in the total score on the upper right.

3.4.1 Bucket Ball Menu:

The game starts with a menu containing general “Play” “Quit” and a “Tutorial” button that gives a brief description of how to play the game.”Hall of fame” button contains a list of top 10 high scores.

3.4.2 Random spawning of balls:

The balls are spawned randomly at the upper part of the window.As the game progresses the speed of the ball gradually increases making it harder and harder to catch them.

3.4.3 Both mouse and keyboard control:

The bucket can be controlled using both the keyboard and the mouse.Pressing ‘A’ and ‘D’ the bucket can be moved left and right.Also by hovering the mouse on the screen the bucket can also be moved left and right giving the user the option to choose whichever they desire.

3.4.4 Sound and Animation:

When a ball is caught with the bucket there is a small lava animation and a sound queue that indicates that a ball has been successfully caught.

3.5 Space Runner

Space Runner is an obstacle avoiding running game.The character runs forward and few obstacles are placed on its way.The main objective of the game is to avoid these obstacles by jumping up and avoiding collision with any of the obstacles.Initially the player has a certain amount of lives.Collision with any obstacle will cause loss of a life.When the player runs out of lives the game ends.Sometime lives are spawned randomly on the map as you run along.Taking them will increase the lives.The score increases for as long as you survive.

3.5.1 Space Runner Menu:

The game starts with a menu containing general “Play” “Quit” and a “Tutorial” button that gives a brief description of how to play the game.”Hall of fame” button contains a list of top 10 high scores.

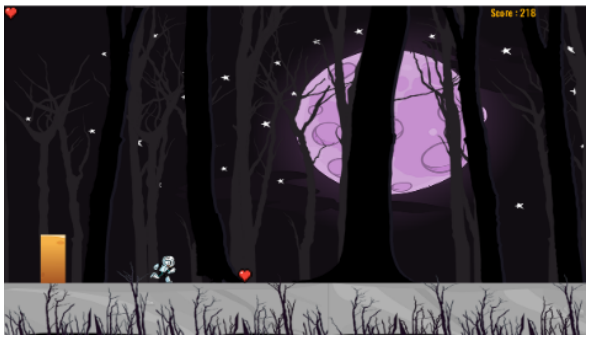
3.5.2 Jumping Mechanics:

The player can jump up using the ‘W’ key on the keyboard.Also if timed properly the character can double jump if he is above a certain height.Obstacles can be avoided by jumping.The character keeps on moving forward while on air to cover horizontal distance.

3.5.3 Running animation and sound:

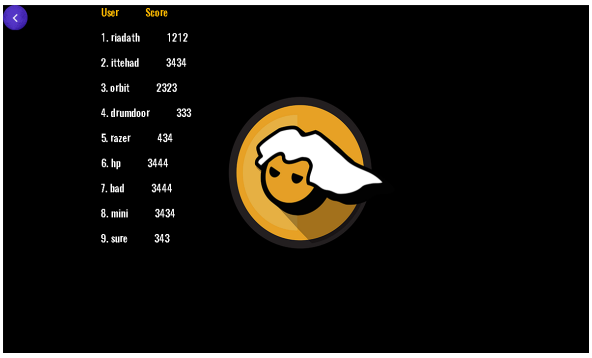
We have created an animation of the character running forward.We also move the background to the left to create the illusion of the character moving forward.In order to make it look more realistic we have added several layers on the background and moved each layer with a different velocity to give a 3D effect to the trees and the sky with the moon.Also a sound queue is added to indicate collision with any obstacle.

3.5.4 Gaining lives in the game:



From time to time a life is spawned in the game giving the player the opportunity to gain more lives and staying alive in the game for a longer amount of time.The player needs to collide with the life inorder to earn it.catching a life in the game will increase the amount of lives on the top left corner of the screen. The player cannot have more than 5 lives.

3.6 High Score Menu



Every game has its own leaderboard. Every game menu has a “Hall of Fame” option. Clicking this menu will prompt the leaderboard. If a player can score higher than any of the top scores currently saved, the player's username will be listed on the hall of fame along with the score.

4. Project Modules

The entire project is designed in a way that every game is coded into a header file. One of the header files that we have used throughout the project is the “texture_timer.h”. It consists of two classes (defined as structures). The “Texture” class and the “Timer” class. They are described below.

texture_timer.h

Texture

Attributes	mTexture : SDL_Texture* mWidth : int mHeight : int	
Methods	loadFile(path:string,ifColKey:bool,r:int,g:int,b:int):bool loadFromText(path:string,textColor:SDL_Color):bool setBlendMode(blending:SDL_BlendMode):void setAlpha(alpha:Uint8):void render(x:int,y:int,clip:SDL_Rect*,flip:SDL_RenderFlip):void getWidth():int getHeight():int	[Load an image file] [Load a text] [Adjust color blending] [Adjust transparency] [Render a texture] [Returns texture width] [Returns texture height]

Timer

Attributes	mStartTime:Uint32 mPausedTime:Uint32 mIfPaused:bool mIfStarted:bool	
Methods	start():void stop():void pause():void unpause():void getTicks():Uint32 isStarted():bool isPaused():bool	[Start the timer] [Stop the timer] [Pause the timer] [Resume the timer] [Get the elapsed time] [Check if the timer is started or not] [Check if the timer is paused or not]

1. Pacman

pacmanLite(username:string):int

Calling this function will run the pacman game. It should return a 0 if it runs successfully.

pacman.h

Functions

loadPacmanMedia():bool	Loads all the media files
load_map():void load_ghosts():void	Loads the map blocks and ghosts positions into arrays to be rendered later on
initPacman():void	Initializes all the global variables
render_ghosts():void render_food():void	Renders all the map objects in the positions previously saved. Spawns a random food and checks if it's valid.If it's valid then it is rendered.If a Food has stayed for a specific amount of time then it is erased.
load_Points():void	Keeps track of the points in the game
checkCollision(a:SDL_Rect, b:SDL_Rect):bool Is_Food_Valid(a:SDL_Rect):bool	Collision detection functions. Checks collision with different map objects.
If_Pacman_Collided_With_Food(a:SDL_Rect):void If_Pacman_Collided_With_Ghosts(a:SDL_Rect):bool If_Pacman_Collided_With_Wall(a:SDL_Rect):bool	Collision detection functions.
pacmanHelpMenuUI(e:SDL_Event):MENU_OPTIONS showPacmanHighScore(e:SDL_Event):MENU_OPTIONS showPacmanScore(e:SDL_Event,username:string):MENU_OPTIONS handlePacmanUI(e:SDL_Event):MENU_OPTIONS	These Functions handle the main menu options which include the leaderboard, tutorial and showing the score after the end of a game.
closePacman():void	Frees all the loaded textures

Structures

PacmanGhost

Attributes	GHOST_WIDTH:static const int GHOST_HEIGHT:static const int upperPos:int lowerPos:int mPosX:int mPosY:int type:int movementType:bool mCollider:SDL_Rect currentClip:SDL_Rect*
Methods	render():void [Renders the ghost at position (mPosX,mPosY)]

PacmanFood

Attributes	FOOD_WIDTH:static const int FOOD_HEIGHT:static const int mPosX:int mPosY:int type:int time:int currentFlip:SDL_Rect*
Methods	render():void [Renders the food at position (mPosX,mPosY)]

Pacman

Attributes	mFlipType:SDL_RendererFlip currentClip:SDL_Rect * PACMAN_WIDTH:static const int PACMAN_HEIGHT:static const int DOT_VEL:static const int mPosX:int mPosY:int mVelX:int mVelY:int frame:int pacmanAnimationSpeed:int mCollider:SDL_Rect

Methods	handleEvent(e:SDL_Event):void [Handles keyboard input] move():void [Moves pacman around the map according to user input] render():void [Renders pacman at position (mPosX,mPosY)]

2. **Icy Tower**

towerGame(string:username):int

Calling this function will run the Icy Tower game. It should return a 0 if it runs successfully.

tower_game.h

Functions:

checkCollision_tower(man :SDL_Rect,plat :SDL_Rect):bool	collision detection function for the tower game
Towerman_Collided_With_Platform(A:SDL_Rect):bool	This function checks if the Towerman has collided with any of the platforms currently present in the tower.
valid_platform(a:int,b:int,c:int,d:int):bool	This function checks if a randomly spawned platform is not impossible to reach from the current platform.
loadTowerMedia():void	Loads all the media files
render_score():void	Renders the current score of the game
load_Tower():void	Loads a random platform and renders it after checking if it's valid.Also removes any platform that has gone below the window and adds new platforms at the top.Also maintains the downward velocity of the platforms.
closeTowergame():void towe_Game_Init():void	Frees all the loaded textures Initializes all the global variables
towerHelpMenuUI(e:SDL_Event):MENU_OPTIONS showTowergameHighScore(e:SDL_Event):MENU_OPTIONS showTowergameScore(e:SDL_Event,username:string):MENU_OPTIONS handleTowergameUI(e:SDL_Event):MENU_OPTIONS	These Functions handle the main menu options which include the leaderboard, tutorial and showing the score after the end of a game

Structures

Towerman

Attributes	mFlipType : SDL_RendererFlip Towerman_WIDTH : static const int Towerman_HEIGHT : static const int Towerman_Vel : static const int jumping_vel : static const int falling_vel : static const int characterFrameDelay : static const int currentClip : SDL_Rect* mCollider : SDL_Rect mPosX : int mPosY : int mVelX : int mVelY : int frame : int jumping : bool prev : bool special : bool
Methods	handleEvent(e:SDL_Event):void [Handles keyboard input] move():void [Moves Towerman around the tower according to user input] render():void [Renders Towerman at position (mPosX,mPosY)]

3. Bucket Ball

bucketBall(*string:username*):int

Calling this function will run the Bucket Ball game. It should return a 0 if it runs successfully.

bucketBall.h

Functions

loadBucketBallMedia():bool	Loads all the media files
initVariable():void	Initializes all the global variables
renderMapBB():void	Renders all the map objects in the positions previously saved.
lavaAnimation():void	Handles the lava animation during collision

checkCollisionBB(bucketShape:SDL_Rect,ballShape:SDL_Rect):COLLISION_TYPE	Collision detection functions. Checks collision with different map objects.
If_Pacman_Collided_With_Food(a:SDL_Rect):void If_Pacman_Collided_With_Ghosts(a:SDL_Rect):bool If_Pacman_Collided_With_Wall(a:SDL_Rect):bool	
bucketHelpMenuUI(e:SDL_Event):MENU_OPTIONS showBucketBallHighScore(e:SDL_Event):MENU_OPTIONS showBucketBallScore(e:SDL_Event,username:string):MENU_OPTIONS handleBucketBallUI(e:SDL_Event):MENU_OPTIONS	These Functions handle the main menu options which include the leaderboard, tutorial and showing the score after the end of a game.
closeBB():void	Frees all the loaded textures

Structures

Bucket

Attributes	mPosX:int mVelX:int mVelocity:int mBucketTexture:Texture mBucketShape:SDL_Rect
Methods	handleEvent(e:SDL_Event):void [Handles keyboard input] move():void [Moves the bucket sideways according to user input] render():void [Renders the bucket] free():void [Frees the loaded texture]

Ball

Attributes	std::mBallSpawnPosition:vector<int> mSPRITE_COUNT:const int mBALL_WIDTH:const int mBALL_HEIGHT:const int mSpawnSpeed:int mEachCatchScore:int mSpawnInterval:int mBallDropSpeed:int mTimer:Timer

	mPrevTime1:Uint32 mPrevTime2:Uint32 mBallTexture:Texture mBallShape:SDL_Rect mBallSprite:SDL_Rect[]
Methods	ballSpawner(bucketShape:SDL_Rect):void [Spawns new ball in random positions] render(bucketShape:SDL_Rect):void [Renders the balls on the screen] free():void [Frees the loaded textures]

4. **Space Runner**

dinoRun(*string:username*):int

Calling this function will run the Space Runner game. It should return a 0 if it runs successfully.

dinorun.h

Functions

loadDinoMedia():bool	Loads all the media files.
renderMapDino():void	Renders all the map objects.
spawnObjects():void	Spawns the obstacles at random positions.
checkCollisionDino(player:SDL_Rect, object:SDL_Rect):bool	Checks if two objects have collided or not.
initVariableDino():void	Initializes all the global variables
dinoHelpMenuUI(e:SDL_Event):MENU_OPTIONS showDinoHighScore(e:SDL_Event):MENU_OPTIONS showDinoScore(e:SDL_Event,username:string):MENU_OPTIONS handleDinoUI(e:SDL_Event):MENU_OPTIONS	These Functions handle the main menu options which include the leaderboard, tutorial and showing the score after the end of a game.
freeDino():void	Frees all the loaded textures

Structures

Player

Attributes	mPlayerTexture:Texture PLAYER_WIDTH:const int PLAYER_HEIGHT:const int PLAYER_VEL:const int PLAYER_SETP:const int JUMP_VEL:const int JUMP_STEP :const int GRAVITY:const int SPRITE_COUNT:const int SPRITE_SPEED:const int BASE_HEIGHT:const int mPosX:int mPosY:int mVelX:int mVelY:int mCurSprite:int mPlayerSprite:SDL_Rect * mFlipType:SDL_RendererFlip mJumping:bool
Methods	handleEvent(e:SDL_Event):void [Handles user input] move():void [Moves the player according to user input] gravity():void [Implements gravity to the game.Changes the Y position of the player if he's above the ground] spriteChanger():void [Changes the spritesheet image to provide the moving animation] render():void [Renders the player at position (mPosX,mPosY)] free():void [Frees the loaded texture]

5. Team Member Responsibilities

We divided the responsibility equally among the both of us.Each of us made 2 of the 4 games that are currently implemented on the game while both of us worked on different parts of the main function.

Reyadath solely wrote and implemented all the codes related to the Bucket Ball game and the Space runner game(dinorun.h and bucket_ball.h).Ittehad solely wrote and implemented all the

codes related to the Pacman game and the Icy Tower game(pacman.h and tower_game.h). As for the main function(main.cpp) Ittehad worked on the designing part of the main map while Reyadath worked on the rest of the coding part.In the texture_time.h header file Reyadath implemented the Texture structure while Ittehad implemented the Timer structure.

Reyadath:

1. dinorun.h

2.bucket_ball.h

3.Texture structure in texture_timer.h

4.Coding part of main.cpp

Ittehad:

1. pacman.h

2. tower_game.h

3. Timer structure in texture_timer.h

4. Designing the main map and handling all the media files for main.cpp

6. Platform, Library & Tools

• Platform: Windows

• Language: C, C++

• Library: SDL2 (Simple DirectMedia Layer), C++ Standard Library

• Tools : Visual Studio Code, Microsoft Paint

7. Limitations

The biggest limitations we faced while making the project was unavailability of enough media resources necessary for the intended idea of the project.Our initial idea was a bit different than the present version of the project we have made.Moving forward we needed to make changes

and adapt as we were unable to find picture/images/figures necessary for the UI and the graphics part of our project that we had initially planned.None of us had any prior experience regarding graphics designing or photo editing before starting this project.We tried to learn a few basic skills to manipulate images we found on the internet and make them suitable for our project.We feel like if we had more experience on graphics designing we could have made our game look much better.

8. Conclusions

What we learned from this project is that development is so much different from the usual problem solving that we have been doing throughout the course. We found that aesthetics of the code is also really important. Following proper conventions allowed us to collaborate better. The biggest difficulty for both of us was that none of us had any prior experience of photo editing or any sort of manipulation with media files. We plan on learning the basics in the future so that we don’t face the same issue again. Overall, this project was a really good experience for us.

9. Future plan

We made this project thinking about all the childhood memories we made playing these classic games and tried to revive them in our project.Although due to limitations of time we were able to implement 4 of our favorite games from quite a long list of childhood games we have made the main structure of our code in a way which enables us to add a new game to our current game very easily.In the future we intend to make more of these games and increase the number of games in our main map.Besides our games have the potential of various future updates and quite a few more features can be added to them.More maps can be added to the Pacman game with increasing level of difficulty.More well designed and creative obstacles could be added to the Space Runner game.An ingame currency can be added to the game that enables the player to play more games to get a higher score.We hope to make our games look nicer and more elegant in the future.

Repositories

GitHub Repository: *https://github.com/riadath/ProjectZ.git*

Youtube Video: *https://www.youtube.com/watch?v=2OyjCgY3Q_E*

References

1.Lazyfoo (SDL Tutorials) : *https://lazyfoo.net/tutorials/SDL/*

2. Official C++ Documentation : *https://en.cppreference.com/w/*

3 .Sprite Resources : The Spriters Resource (spriters-resource.com)