# CS 224 - Object Oriented Programming
# Project Synopsis

Ali Asghar Kerai - ak06857
Areeb Adnan Khan - ak06865
Muhammad Jazzel Mehmood - mm06886
Lakshman Maheshwari - lm06435

$11^{th}$April 2022

## Bhuruz

Bhuruz is a obstacle tackling survival game in which the player has to tackle obstacles while moving through a cylindrical tunnel as a bike type object. The start-end mechanism of the game depends on the health of the player which decrements and increments in a few discrete levels and a health bar corresponding to the status of the player's health is displayed on the screen. The health decreases when the player hits the obstacles which they are supposed to dodge. If the player hits the obstacle of the "bomb" class, the game ends immediately irrespective of the available health of the player. The game ends when the player runs out of health. The objects that the player might encounter can be of varying sizes as well, and their size corresponds to the damage they will incur in the player's health if the collision occurs.

To make the game interesting and to give the player a chance to prolong their game play, health increment object will also appear randomly in the path and if the player collides with those object, there will be an increment in the player's health. The score will be displayed on the screen alongside the health bar to indicate the progress of the player. Assisting the player to navigate through the game, there will be 4 screens which include the Game Start screen, the live game screen, the Game End screen and the credits screen.

Also, there are 3 levels of the game namely Easy, Medium and Hard. The player can choose to play either of the levels before the game begins. The difficulty levels will be based on difference in the speed with which the player will encounter obstacles while playing the game.

The player interacts with the game using the arrow/WASD keys and the mouse, in order to minimize the number of keys being used for a better user experience.

To implement this game as the course project all the techniques of OOP will be applied namely classes and objects, abstraction, encapsulation, inheritance and polymorphism. To improve the memory efficiency all/most of the objects will be created dynamically. SDL 2.0 graphic library will be used to render the graphics for this game, which is necessary to make the game interactive.

# Assets

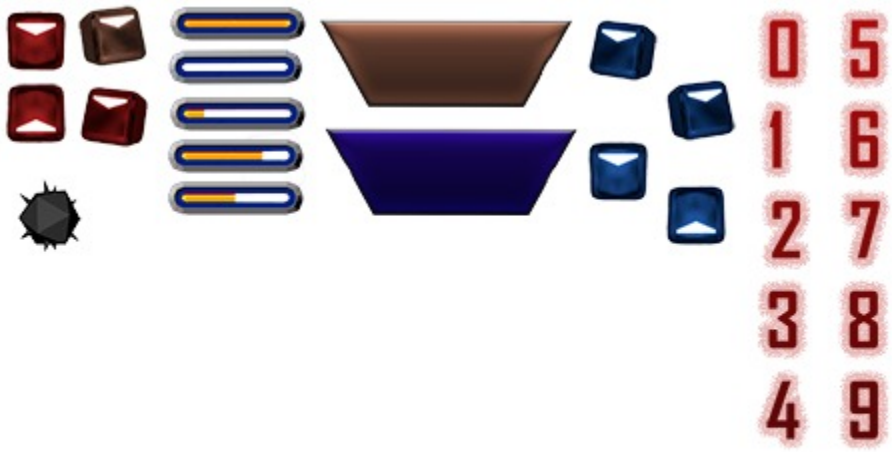



Figure 1: Assets - Obstacles, Score, Health

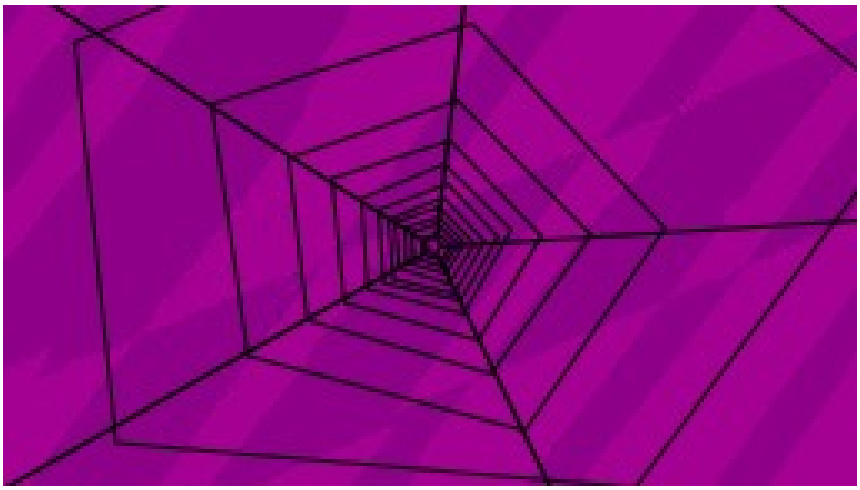


Figure 2: Assets - Background 1

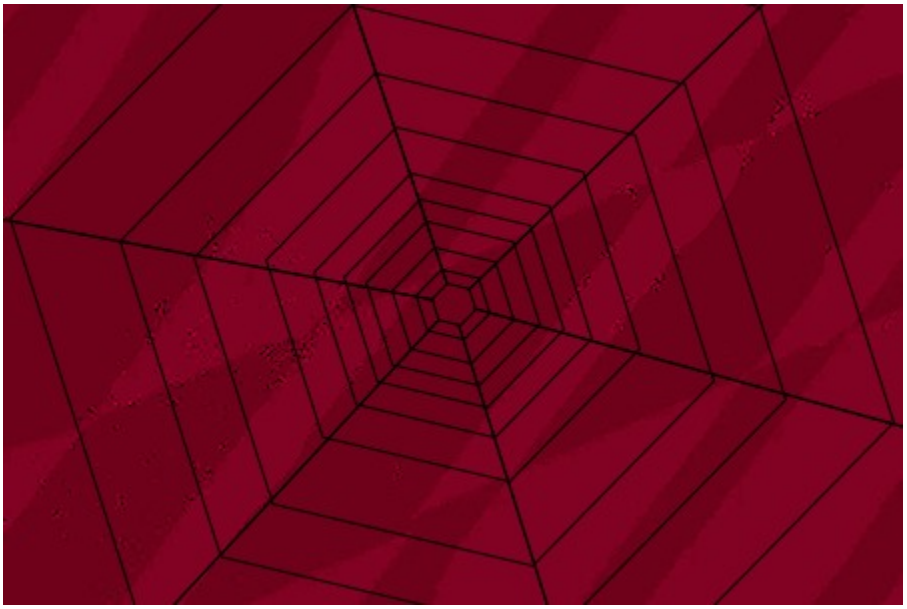
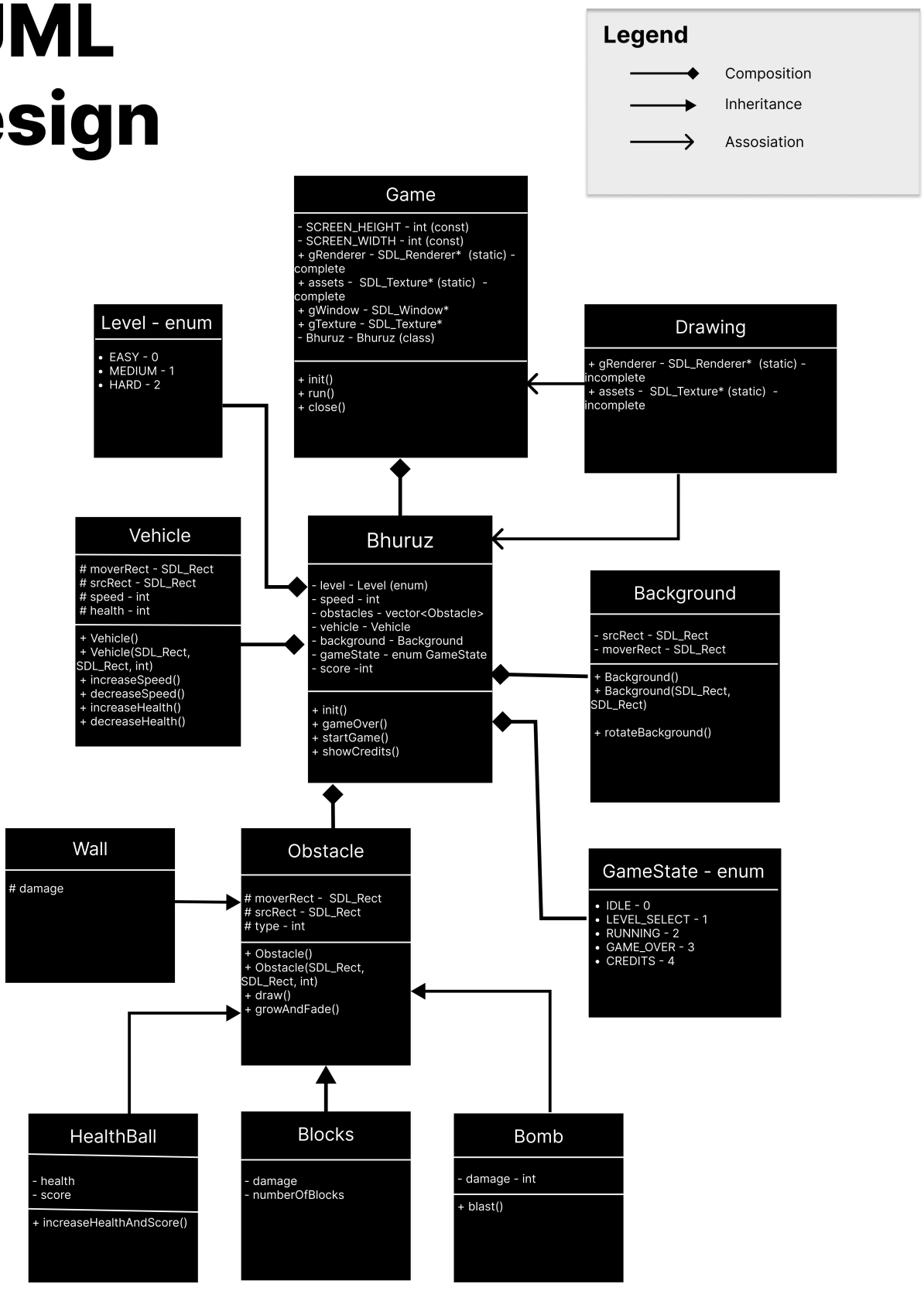


Figure 3: Assets - Background 2

Figure 4: Assets - Main Screen

# UML Design


Legend
Composition
Inheritance
Assosiation
Game
- SCREEN_HEIGHT - int (const)
- SCREEN_WIDTH - int (const)
+ gRenderer - SDL_Renderer* (static) - complete
+ assets - SDL_Texture* (static) - complete
+ gWindow - SDL_Window*
+ gTexture - SDL_Texture*
- Bhuruz - Bhuruz (class)
+ init()
+ run()
+ close()
Level - enum
• EASY - 0
• MEDIUM - 1
• HARD - 2
Drawing
+ gRenderer - SDL_Renderer* (static) - incomplete
+ assets - SDL_Texture* (static) - incomplete
Vehicle
# moverRect - SDL_Rect
# srcRect - SDL_Rect
# speed - int
# health - int
+ Vehicle()
+ Vehicle(SDL_Rect, SDL_Rect, int)
+ increaseSpeed()
+ decreaseSpeed()
+ increaseHealth()
+ decreaseHealth()
Bhuruz
- level - Level (enum)
- speed - int
- obstacles - vector<Obstacle>
- vehicle - Vehicle
- background - Background
- gameState - enum GameState
- score -int
+ init()
+ gameOver()
+ startGame()
+ showCredits()
Background
- srcRect - SDL_Rect
- moverRect - SDL_Rect
+ Background()
+ Background(SDL_Rect, SDL_Rect)
+ rotateBackground()
Wall
# damage
Obstacle
# moverRect - SDL_Rect
# srcRect - SDL_Rect
# type - int
+ Obstacle()
+ Obstacle(SDL_Rect, SDL_Rect, int)
+ draw()
+ growAndFade()
GameState - enum
• IDLE - 0
• LEVEL_SELECT - 1
• RUNNING - 2
• GAME_OVER - 3
• CREDITS - 4
HealthBall
- health
- score
+ increaseHealthAndScore()
Blocks
- damage
- numberOfBlocks
Bomb
- damage - int
+ blast()