Games in C++ – Assignment Exercise 1 – Planning Evidence document
Recreating 'Pong'
Dominik Heiler - 23015707

For our first main assignment in the 'Games in C++' module, we were tasked with coding our own version of the arcade game 'Pong', using C++ and the accompanying SFML code library.

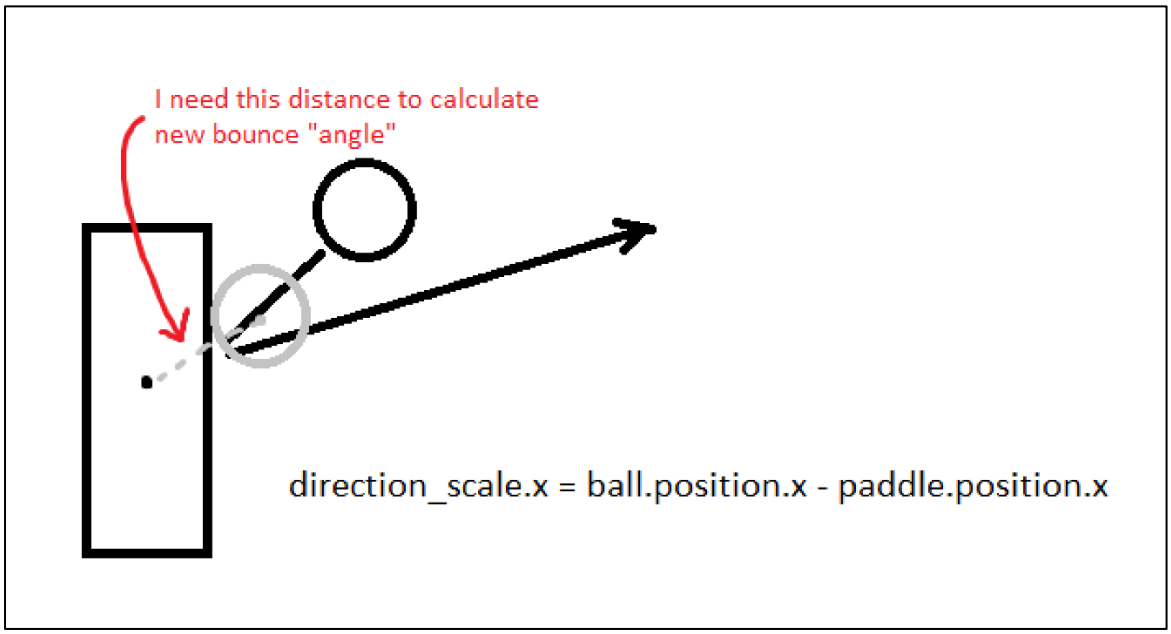
Github Username: DominikHHH

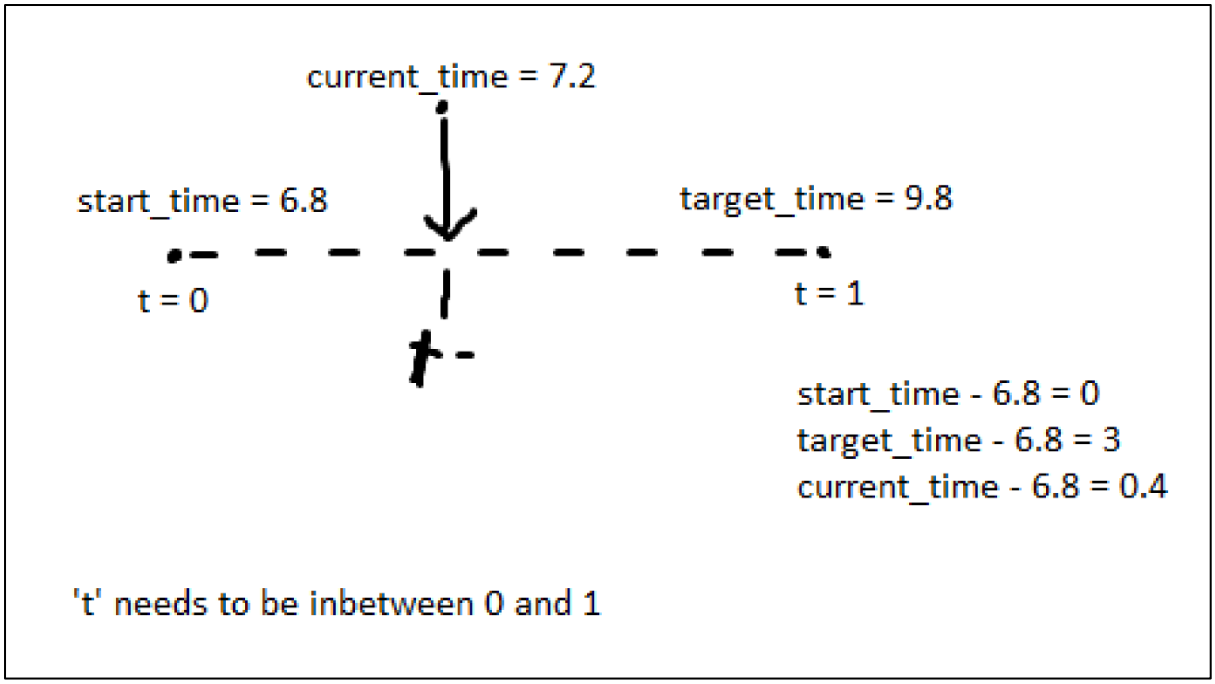
Link to Source Code: https://github.com/UWEGames-GiC/pong-23-24-DominikHHH

Presented below are some of the planning materials that I made for myself while coding the actual game assets themselves.

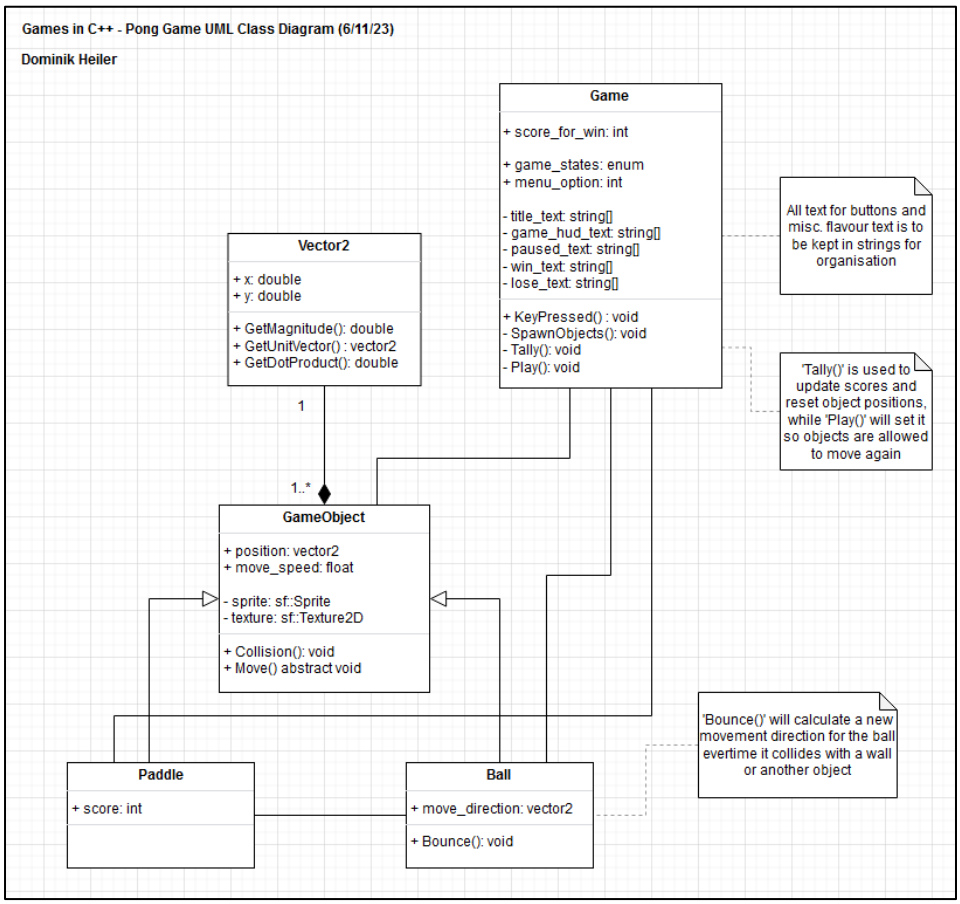
```
Games in C++ - Pong Game - Notes for Base Plans
Dominik Heiler

-----------------------------------------------------------

Main game objects:
 - Paddle
   - Player Paddle (paddle1)
   - AI Paddle (paddle2)
 - Ball

UI objects:
 - Titlescreen:
   - Title Text/Graphic
   - Menu Option Texts/Graphics
 - Main game:
   - Scoreboard Text
     - Two on both sides of the "arena"
   - 'Player 1' and ' Player 2' nametags
 - Pause screen:
   - 'Paused' Text/Graphic
   - Menu Option Texts/Graphics
 - Game over screen:
   - "Game Over" Text/Graphic
   - Menu Option Texts/Graphics

-----------------------------------------------------------

Controls:
 - AD Keys - Menu navigation
 - WS Keys = Player 1 controls
 - Up/Down Arrow Keys = Player 2 controls
 - Enter - Pause/Quit

-----------------------------------------------------------

Personal notes:
 - Collision detection to use regular AABB
   - Adjusting the angle of the ball could be done by comparing dot product maybe?
   - Angle variable should NOT be an angle or else the sprite's appearance and collision might break
 - Two separate colours for paddles
 - Enums for menu navigation maybe, instead of switches
 - Could make additional modes later if I get the time
        - Player VS AI
        - Player VS Themselves? (Paddle orbits around the centre of the screen, constantly changing the
          paddle sprite's rotation in 'update')
```

```
Games in C++ - Pong Game - Pseudo-Code Collection
Dominik Heiler

-----------------------------------------------------------

Simple collision detection

bool Collision (object_a, object_b)

        bool colX, colY

        // Horizontal collision
        if (object_a.x + object_a.size.x) is bigger than (object_b.x)
        AND if (object_b.x + object_b.size.x) is bigger than (object_a.x)
                colX = true

        // Vertical collision
        if (object_a.y + object_a.size.y) is bigger than (object_b.y)
        AND if (object_b.y + object_b.size.y) is bigger than (object_a.y)
                colY = true

        if both colX and colY equal true
                return true


-----------------------------------------------------------

Ball bouncing logic

if ball is colliding with ball

        ball.velocity.x *= -1
        ball.velocity.y *= dotProduct(paddle.normal, ball.velocity)

The ball is going to get mirrored no matter which way it's going to get hit, so i just need to multiply by -1
'paddle.normal' would be the right-side face of the paddle (with a value of (0, 1))

-----------------------------------------------------------

Menu navigation

enum game_states = 'TITLE_SCREEN', 'MAIN_GAME', 'PAUSE', 'GAME_OVER';
int menu_choice; (Choosing between wanting to replay the game or quit can just be
                 changed with an int since I think I will change them in dedicated functions)

-----------------------------------------------------------
```

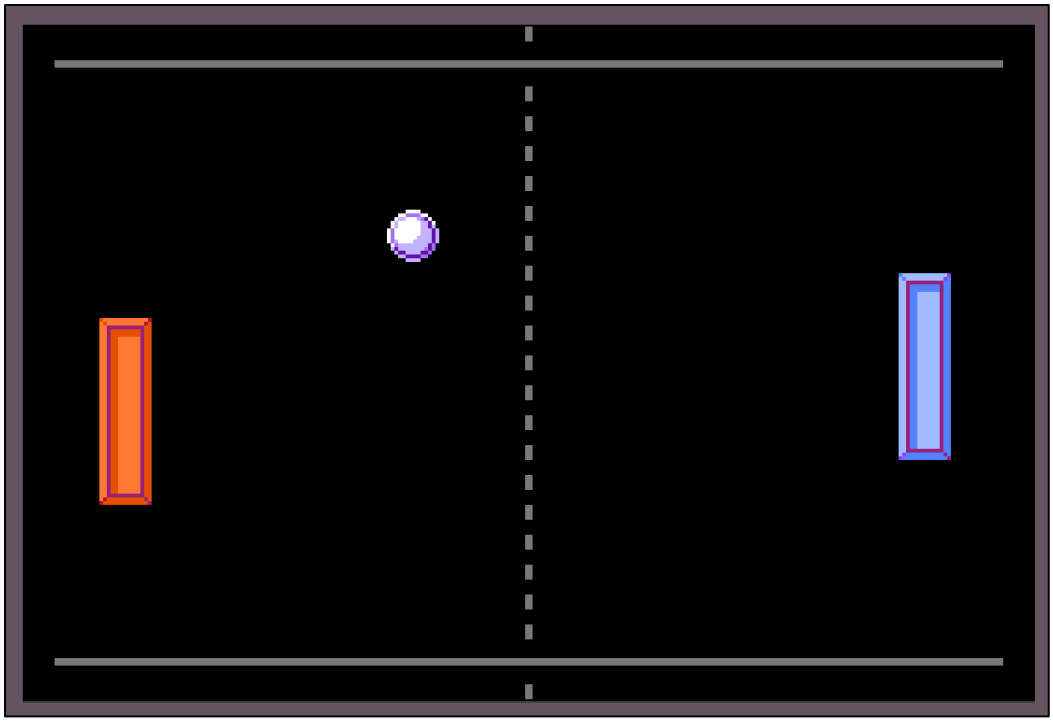




(Above: screenshots of my notes, pseudo-code snippets, and doodles that I made for myself at the start of, and during, the project's development, having used 'Notepad' and 'MSPaint' for future ease of access. The doodles, in particular, were made just for myself as a way to keep track of what I needed to work on, so they are quite rough and quickly-produced)



(Above: the main UML class diagram that I made for myself using 'draw.io', having planned out all the main object classes that I will use in my game)



(Above: a basic mock-up featuring some simple visuals that I drew in ‘Aseprite’)

C++ code and graphical assets made by Dominik Heiler

Software used: CLion, Gitkraken Aseprite, draw.io

‘Press Start 2P Font’ by codeman38 (https://www.fontspace.com/press-start-2p-font-f11591)