Nacimo Jama

NEA – 2D GAME

Contents

[Over – View 1](#_Toc151367162)

[Analysis 2](#_Toc151367163)

[Features to make problem solvable 2](#_Toc151367164)

[Research 4](#_Toc151367165)

[features to identify solutions to computational solutions 6](#_Toc151367166)

[Limitations 7](#_Toc151367167)

[Requirements for solutions 7](#_Toc151367168)

[Success Criteria 9](#_Toc151367169)

[Design 10](#_Toc151367170)

[Structure 11](#_Toc151367171)

[Solution Algorithms 11](#_Toc151367172)

[Usability Features 11](#_Toc151367173)

[Key Variables, Data Structures, Classes & Validation 11](#_Toc151367174)

[Test Data 11](#_Toc151367175)

[Further Post Development Data 11](#_Toc151367176)

[Development Of Coded Solution 11](#_Toc151367177)

[Iterative Development Process 12](#_Toc151367178)

[Evidence Of Prototype Versions 12](#_Toc151367179)

[Modular & Well-Structured Solution 12](#_Toc151367180)

[Annotations 12](#_Toc151367181)

[Variables and structures 12](#_Toc151367182)

# Over – View

* For my NEA project, I will be creating a classic game which is similar to flappy bird and Mario and utilising features from other similar 2D games such as Crossy road. I will be creating a one player game where the character would be navigating through obstacles and will be offering players coins to collect and challenges and missions to perform while playing to increase the game's replay value. These aims give objectives which are beyond merely just avoiding obstacles. The game ends when the player collides with an obstacle and the player can choose whether they want to restart the game or quit the program.

## Analysis

### Features to make problem solvable

• Described and justified the features that make the problem solvable by computational methods, explaining why it is amenable to a computational approach

Computational thinking

Why use a computer?

* To create the game and write my coursework, it will be necessary to have a computer since there aren’t any other ways in which I can complete my project. Furthermore, a computer is an amenable approach to my game since the game needs to be infinite until the player dies which is not possible on a board game as board games have a certain point where it stops.

Debugging:

* I’ll be utilising debugging tools in my program as it will allow me to identify any errors that needs to be fixed which will ease the process of coding. This will also ensure that the game runs perfectly without any errors for the stakeholders.

Abstraction:

* I’ll be removing unnecessary real-world elements into simplified obstacles. This abstraction of simplified graphics will help to keep the focus on the gameplay and makes it visually appealing. I’ll be doing this since implementing 3D graphics would require more time than I have on this project, and it would be a waste of time as my stakeholders will prefer to have a 2D game rather than a 3D.
* I’ll abstract the complexities of real-world gravity for my 2D game since my character will be flying by using the up and down keys which will allow me to make it accessible to a wide range of players. Furthermore, simplifying the movements of my characters to make my code easier to write will be advantageous since I’m limited on time. In addition to this, the mechanics that the player can use will be abstracted since pressing the ‘W.S.’ keys whilst moving the characters wouldn’t reflect the actual movements in real-life.
* I’ll abstract the HD quality of my 2D game by making it more pixelated and keeping the minimal art style which gives a sense of nostalgia and retro look for players of all age to play.
* By removing real world elements from my 2D game, I’ll be able to add a moving background to create the illusion of constant movement and it allows the player to focus on navigating the character through the obstacles without worrying about controlling the entire game.

Pattern Recognition:

* I will be using repetitive object movement and collision detection as a form of pattern recognition as it allows me to reuse the movement of the moving clouds/any other obstacles in my game as this is appropriate for my stake holder since it makes the game play easier.
* Pattern recognition will be used for collision detection in my program as the constant up and down movements of the clouds can be used so that the algorithm can check the players position to see if it has interacted with the clouds thus determining the outcome of the game.

Decomposition:

* I’ll be using decomposition in my 2D game by breaking down the player movements into smaller components as it creates more efficient and responsive gameplay for the stakeholder.
* I’ll be using subroutines to code my game as a decomposition method as I’ll be able to break down large components into smaller ones to make it easier to correct mistakes and test them which will save time. For example, I’ll break down the movements such as flying up or down into subprograms like flyUp() and flyDown().,
* For the high score table, I’ll utilise decomposition by implementing an insertion sort to store all the high score of players by rank.

Stakeholder

• Identified suitable stakeholders for the project and described them explaining how they will make use of the proposed solution and why it is appropriate to their needs.

Stakeholder = My younger brother, a year 6 student in primary school right now, and anybody around that age. He enjoys playing these classic video games with easy-to-understand controls that give players tasks and objectives to do while having fun. My game is suitable for my stakeholder as it is not graphic thus age appropriate. My stakeholder will make use of this project since my game will have goals and objectives, which will encourage them to play more frequently.

## Research

• Researched the problem in depth looking at existing solutions to similar problems, identifying and justifying suitable approaches based on this research.

Similar software applications to my chosen project:

1. Flappy bird
2. Super Mario Bros

The mechanics of how Flappy bird have been made to work:

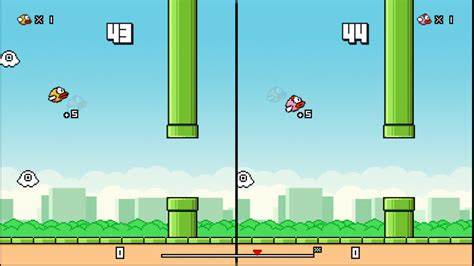
* Flappy bird operates on a continuous game loop in which the game’s state is changed and repeatedly rendered. This loop frequently employs a predetermined frame rate (such as 30 or 60 frames a second) to guarantee fluid gameplay.
* Flappy bird includes a scoring system where the player earns points for passing through the gaps in the game without any collisions being detected and the game tracks the number of pipes the player has navigated through giving them their score at the end.

Scoring system:

Scoring higher scores in the game creates a sense of accomplishment in players which enhances the overall enjoyment of the gaming experience for the players.

Number of lives:

Being aware of the limited number of lives allows the players to find themselves more deeply engaged and focused resulting in a more intense and gratifying gaming experience.



Moving pipes:

The ongoing motion of these pipes creates a dynamic and constantly evolving challenge for players – the unpredictable nature of their movement allows the players to adapt swiftly thus preventing the game from becoming dull.

Character jumping

* This game utilises audio by adding basic sound effects like when the bird’s wings flap when it jumps and when the bird collides with its surroundings as it navigates through the gaps in the pipes.
* This game checks to see if the bird character has any collision with its surrounding thus determining the outcome of the game.

Features and approaches used in flappy bird:

* Flappy bird utilises minimal soundtrack that includes repeating soundtrack and basic sound effect which is a simple feature that contributes to the retro aesthetics alongside the pixelated and minimal art style which gives a sense of nostalgia and retro look.
* Flappy bird utilises features which are relatively straightforward such as the simplicity of the controls like the one-tap control scheme; the players of flappy bird just need to tap the bird in order to make it flap its wings and go through the obstacles, contributing to its accessibility and addictiveness.

Basic movements:



## In developing my game, I've observed the established convention in 2D plat formers like Super Mario Bros, where players navigate from left to right across the screen. Following this standard, I plan to implement a side-scrolling mechanic where the character moves horizontally while the background scrolls in the opposite direction, creating a seamless illusion of movement.

## Additionally, inspired by Super Mario Bros, I intend to incorporate collectible coins scattered throughout the game environment. These coins will offer players advantageous features or rewards, enhancing the gameplay experience and adding depth to the mechanics.



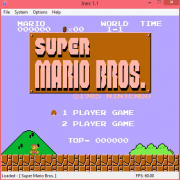
Implementing a jump mechanic is an essential feature in any platformer game, serving as the core action enabling traversal and interaction. Recognising the variation in jump types and physics across platformers, I aim to add this feature into my game to enhance player interactivity. I will do this by developing a function specifically dedicated to jumping, thus allowing the players to have diverse opportunities for collecting items and engaging with the game environment, enriching their overall experience.

Visual interface:

## Ultimate Super Mario Bros Game Free Download

In Super Mario Bros, players must keep track of key values like score, remaining lives, and time spent. These values are typically displayed on screen in a prominent yet non-intrusive location, often at the top, bottom, left, or right, ensuring players can easily monitor them without diverting attention from the main game. To ensure readability, the numbers are commonly presented in white with a black outline, ensuring clarity against varying backgrounds. This design choice is crucial as clear text is vital for players to understand their progress and actions within the game. It's a standard practice across most games, including mine, to prioritize readability for an optimal gaming experience.

Main menu:



* Creating a game menu is really important for players to have a good time. It should look nice and show all the information clearly. In the example above, the background is kept simple so it doesn't look too crowded. The words are in white to stand out against darker backgrounds. The menu doesn't have too many options to avoid confusing players. Buttons are named clearly so players know what they do. If players don't get it, hovering over a button shows a short description.
* In my game, having well-named and user friendly buttons is super important because it helps players understand what each button does, no matter how it looks.

## features to identify solutions to computational solutions

• Identified the essential features of the proposed computational solution explaining these choices

* One essential feature I would implement in my game is using the ‘W.A.S.D’ keypads on the keyboard to allow my stakeholders to move up, down, left and right therefore a computer is amenable for this as on-board games, you cannot make jumping/running/ flying movements.
* Another essential feature that I’d add into my game is for my characters to fly and navigate through obstacles allowing my stake holders to have an interactive gameplay since this will allow them to progress through the game and score high in the high score table.
* Another feature I would add to my game is a high score table. This is because this will be a requirement in my game since my stakeholders would want to know where they are on the score table.
* A computer is an amenable approach as the game needs to be infinite until the player dies which is not possible on a board game as board games have a certain point where it stops.

## Limitations

• Identified and explained with justification any limitations of the proposed solution.

* Sound - the sound of character movements will be taken from an existing project since I don’t have enough time to create sounds for my project.
* Time – it will be difficult to add many levels to my game since I’m limited on time, so I’ll only be able to make 1 level within the time limit I have.
* Multiplayer- I will not be able to add many players to my game since I don’t have much experience in that area, so this game will only be a one player game.

## Requirements for solutions

• Specified and justified the requirements for the solution including (as appropriate) any hardware

and software requirements.

Hardware Requirements:

|  |  |  |
| --- | --- | --- |
|  | Hardware | Justification |
| 1 | Speaker | User would need this since they would need to hear the sound effects but it’s not a must |
| 2 | Monitor | Need the screen in order to view the game |
| 3 | Computer | Need this to create my video game and this project can’t be completed without a computer. The user would also need a computer in order to play my game. |
| 4 | Keyboard and mouse | Keyboard- Use this to use the ‘W.A.S.D’ keys to control character  Mouse- navigate through the game |

Software Requirements:

|  |  |  |
| --- | --- | --- |
|  | Software | Justification |
| 1 | Word document | Write down my coursework |
| 2 | Visual studio | Write down my code |

Design Requirements:

* User interface requirements
* Functionality requirements
* Level requirements
* Extra requirements

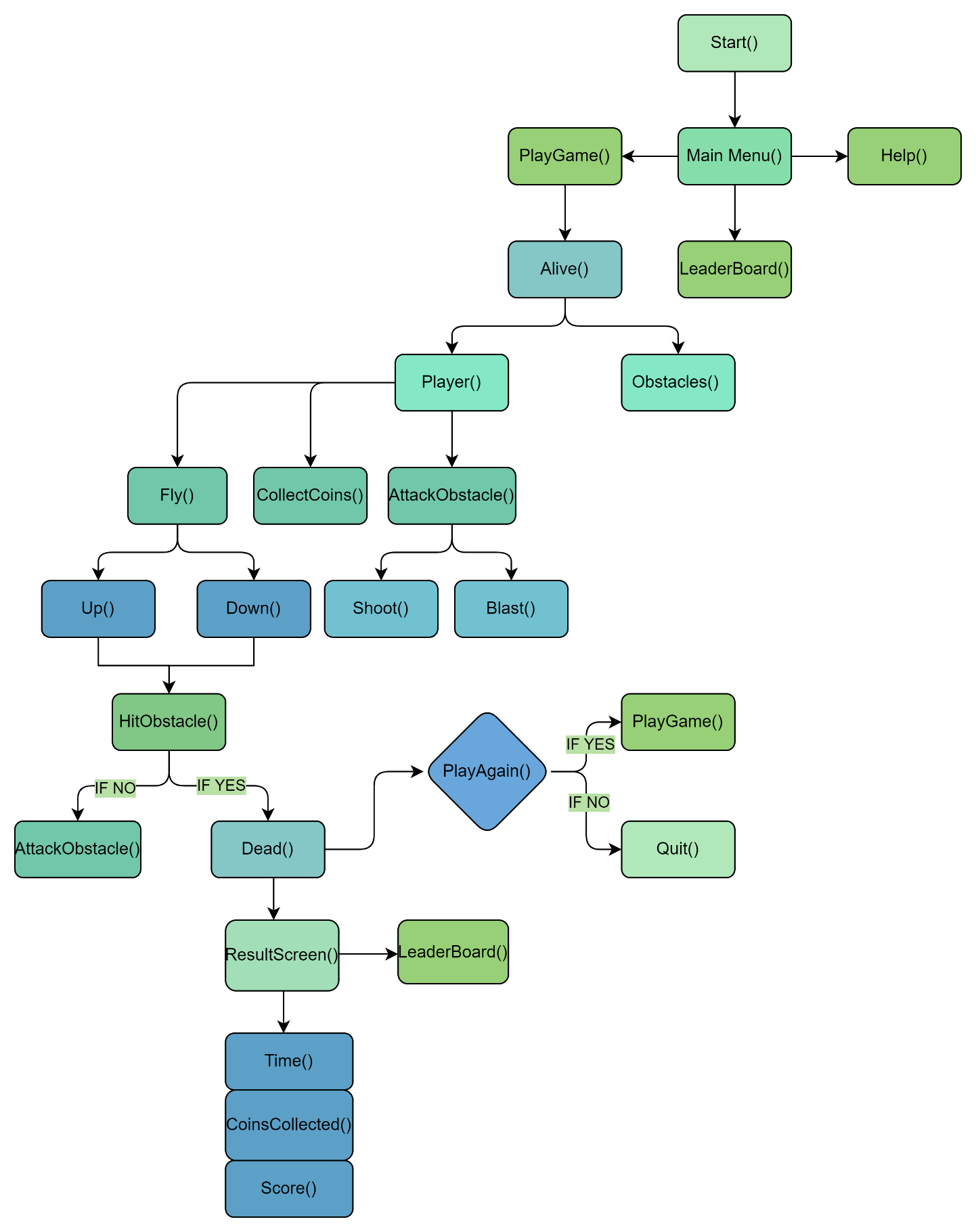
|  |  |  |
| --- | --- | --- |
|  | Requirements | Justification |
| 1 | Game displays in full screen | Enhances the users experience and makes it more enjoyable for them with no distractions |
| 2 | Timer | Tells the player how long they were playing for |
| 3 | Main menu splits into 3: play game, high score table, options | Tells players what each button does |
| 4 | Character can fly/jump | player needs to be able to fly/jump through obstacles in order to survive. |
| 5 | Character can move up and down | player needs to be able to move the character |
| 6 | Character loses when they touch an obstacle | This will add difficulty to the game since the player will have to be more careful when getting near an obstacle |
| 7 | Character can collect coins | This will add an objective to the game making it more fun for my stakeholders. |
| 8 | A clear end to the level – when this happens, game will show result screen | The result screen will display the player’s time, score and rank on leader board. |
| 9 | The level has appropriate music playing | Creates an atmosphere making it more enjoyable for the player |
| 10 | Characters, coins have sound effects | Makes the user experience more enticing for the player |
| 11 | Internet connectivity | Needed for me to complete my project and for the player to play the game |

## Success Criteria

• Identified and justified measurable success criteria for the proposed solution

|  |  |  |
| --- | --- | --- |
|  | Success Criteria | Justification |
| 1 | When space is pressed the character jumps/fly | Player needs to jump/fly |
| 2 | There is music playing | There needs to be music to make an atmosphere |
| 3 | Play game button needs to be working when clicked | This button needs to work for the player to play the game |
| 4 | Player loses the game if they collide to an obstacle | Player needs to make sure they don’t collide to an obstacle or else they will lose the game. |
| 5 | When player pressed the ‘W’ key, the character moves up | Player needs to be able to move up |
| 6 | When player pressed the ‘S’ key, the character moves down | Player needs to be able to move down |
| 7 | Each screen has a unique image | Helps user differentiate between different screens |
| 8 | Result screen displays after the level ends | Needs to be end to the level |
| 9 | The level scrolls down when the player moves up. The level scrolls up when the player moves down. | The level needs to match the characters movements |
| 10 | The user should be able to view the high score table | They should be able to compare themselves to their previous performance |
| 11 | Character should be able to collect coins and the coin count should increase on the screen | Player should be able to see how much coins they’ve collected |
| 12 | There should be a menu | The player must be able to navigate through the game |

## Design

Decomposition

**Explanation of this Process**

**The development of my game will be much easier by breaking down the game into several components. Decomposition allowed me to do this as it useful for programming objects into much smaller elements that will be easier to program and test.**

**Furthermore, the player will be the most important object in my game as it is responsible for all the actions that the stakeholders will make so by focusing on this, my end prototype will look much more interactive for my stakeholder.**

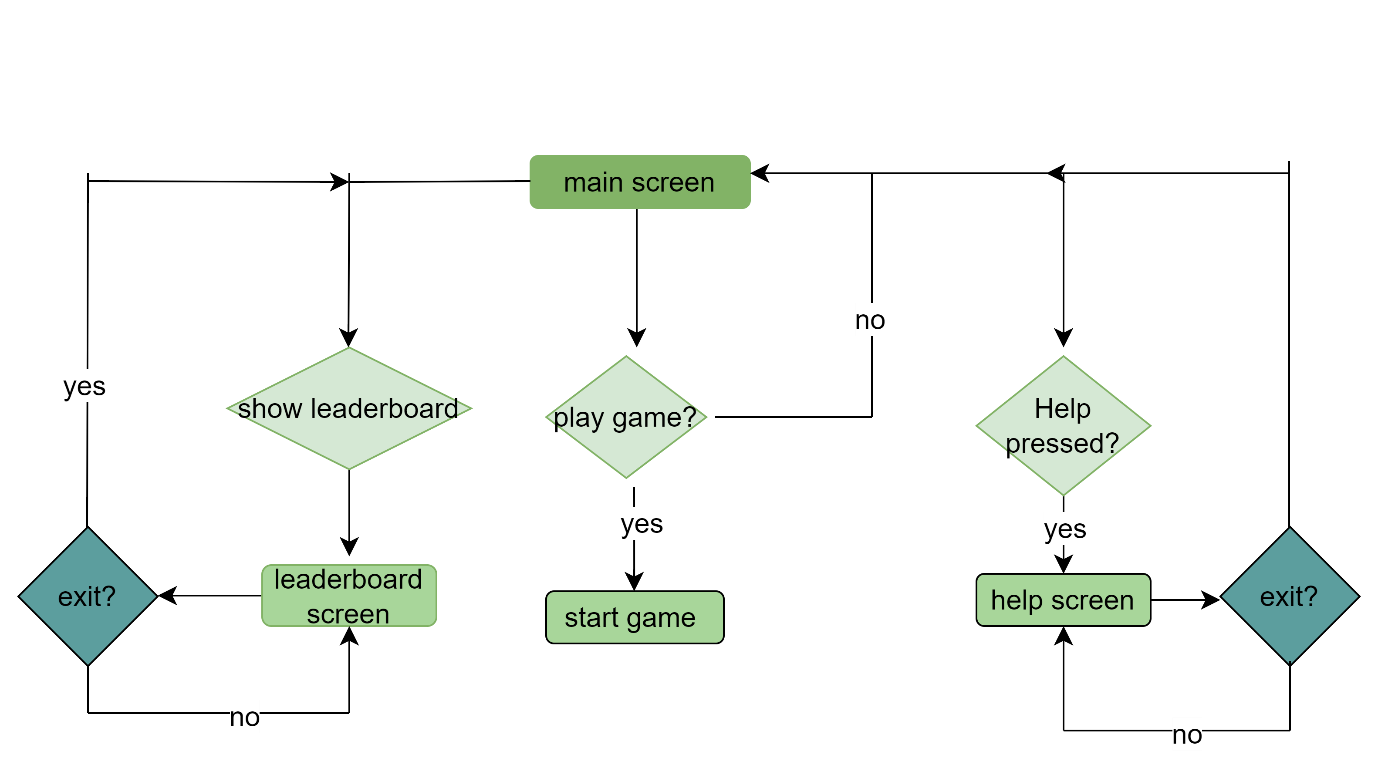
**In addition to this, the breaking down of these objects will keep my game modular in nature since I broke each of the objects down further allowing me to spend less time writing the code.**

**Structure**

* Define in detail the structure of the solution to be developed.

Main Menu:

Flowchart showing how menu is navigated through and accessed

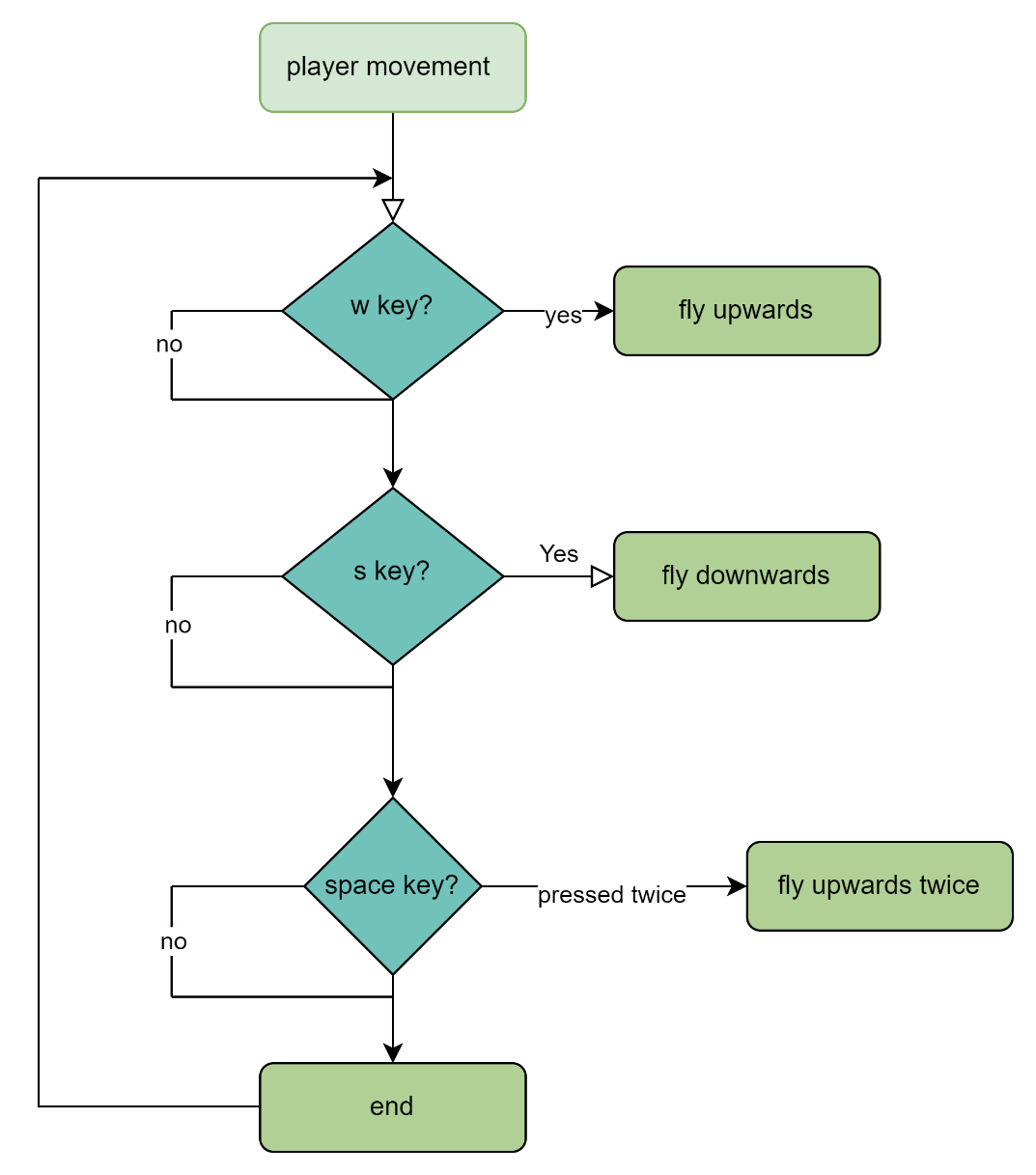


Leader Board:



 Flowchart shows how leader board will be accessed and navigated through.

Player:



Flowchart shows player movements and the keys to be pressed to move the character.

Solution algorithms:

* Describe the solution fully using appropriate and accurate algorithms justifying how these algorithms forma a complete solution to the problems.

Character Movement:

If (Key == ‘A’ pressed)

Left = true

If (Key == ‘D’ pressed)

Right = true

If (key == Spacebar && Jump == false)

Jump = true

**Justification:**

The 'A' and 'D' keys are often used for left and right movement in games.

The spacebar is commonly used for actions like jumping in plat former-style games.

The conditions for the spacebar check ensures that the character can only jump if they are currently not in the air, preventing double jumps or continuous jumping while already in the air.

Character movement if the above keys are not pressed:

**Justification:**

If (key == A)

Left = false

If (Key == D)

Right = false

If (Jump == true)

Jump == false

Releasing the 'A' key (`Left = false`) and 'D' key (`Right = false`) indicates that the player is no longer holding these keys for movement, so the corresponding movement actions should stop. This ensures that the movement actions are only active while the keys are pressed.

Resetting `Jump` to false (`Jump = false`) when `Jump` is true ensures that the character has completed a jump. This again prevents continuous jumping, ensuring that the character can only jump again after landing.

Character movement – JUMPING:

**Explanation:**

If (jumping is true and force is < 0) THEN

Set jumping to false.

If jumping is true:

Set Speedjump to -12.

Decrease force by 1.

Else:

Set Speedjump to 12.

If the character is jumping (`jump` is true) and force is less than 0, indicating the character is coming down, set `jump` to false to end the jump.

While jumping (`jump` is true), set `Speedjump` to -12 for upward movement and decrement `force` to simulate gravity.

If not jumping (`jump` is false), set `Speedjump` to 12 for downward movement.

**Justification:**

* Manages character's jumping state for smooth gameplay.
* Simulates realistic jumping and falling with speed adjustments.
* Crucial logic for platformer-style game movement.

Character movement – RIGHT:

**Justification:**

If (Go\_Right is true AND (Bird's position + (Bird's width + 60)) is less than the width of the screen) THEN

Move Bird to the right by the value of SpeedBird.

Set Bird's image to the right-facing sprite.

Enables responsive player control for moving the character right.

Maintains game boundaries to prevent character overflow.

This also provides visual clarity with right-facing sprite feedback whilst also enhancing player immersion and game graphics.

Character movement – LEFT:

**Justification:**

If (Go\_left is true AND Bird's Left position is greater than 60):

Decrease Bird's Left position by SpeedBird.

Set Bird's Image to the left-facing sprite.

This ensures responsive controls for leftward character movement and Enhanced immersion through clear visual feedback when the left-facing sprite appears.

I implemented boundary constraints to prevent off-screen movement which improves the overall gameplay experience.

Detect Collisions:

Treasure:

If (Bird collides with the Treasure):

If (CoinsCount is >= to 10):

Play the treasure collected sound effect.

Remove the Treasure from the game.

Set HasTreasure to true.

Else:

Set HasTreasure to false.

**Explanation:**

Checks if the player has collected at least 10 coins.

If true, plays a sound effect, removes the treasure, and sets HasTreasure to true.

If false, sets HasTreasure to false.

This facilitates game progression based on coin collection and enhances player engagement through goal-oriented gameplay.

**Justification:**

* Enforces game rule: Collect coins to obtain treasure, adding depth and challenge.
* Provides feedback: Sound effect and removal of treasure signify goal achievement.
* Manages game state: Updates HasTreasure to track player progress.
* Enhances engagement: Encourages exploration and interaction for player satisfaction.

Portal:

**Justification:**

If (Bird collides with the closed portal AND the player has the treasure)

Switch the closed portal image to an open portal.

Stop the game timer.

Show a message box indicating level completion.

Restart the game.

By checking if the Bird collides with the closed portal (`portalClose`) and ensuring that the player has the treasure (`HasTreasure`), the code signifies that the conditions for completing the game have been met. This adds a clear objective for players to strive towards, enhancing gameplay depth and satisfaction.

Visual Feedback: Changing the image of the closed portal to an open portal visually indicates to the player that they have successfully completed the level. This provides immediate feedback, reinforcing the sense of accomplishment and progression.

Game Timer Management: Stopping the game timer (`GameTimer.Stop()`) freezes the game state, preventing further gameplay actions and signalling the end of the current level. This ensures that players cannot continue interacting with the game until they choose to restart.

Game Progression: Restarting the game (`RestartGAME()`) prepares the game environment for players to replay the current level if they wish. This ensures smooth progression through the game and encourages players to continue their journey.

Coins:

**Explanation:**

For each control x in the game screen:

If x is a PictureBox and its tag is "Coin":

If Bird collides with x:

Play coin collected sound effect.

Remove x from the game screen.

Increment CoinsCount.

The loop checks for collisions between the player character (referred to as "Bird") and coins on the game screen. If `x` is a `PictureBox` with the tag "Coin" and collides with Bird, it plays a sound effect, removes the coin from the screen, and increments CoinsCount.

**Justification:**

* Feedback: Provides auditory and visual feedback to the player upon collecting a coin.
* Interaction: Handles collision detection and coin collection efficiently.
* Scorekeeping: Updates the CoinsCount variable to track player progress.

Platform:

For each control x in the game screen:

If x is a PictureBox with the tag "platform" and there is a collision between Bird and x:

Set force to 9.

Move Bird's top position to the top of x minus Bird's height.

Set Speedjump to 0.

**Explanation:**

This loop iterates over each control on the game screen. For controls that are `PictureBox` objects with the tag "platform" and intersect with Bird's bounds, it performs the following actions:

1. Set the `force` variable to 9, likely resetting

2. Position Bird's top to the top of the platform (`x`) minus Bird's height. This places Bird on top of the platform after collision.

3. It sets `Speedjump` to 0, to halt Bird's upward movement after landing on the platform.

**Justification:**

* Collision Handling: This loop efficiently handles collisions between Bird and platform controls, ensuring that Bird behaves appropriately when interacting with platforms.
* Physics Adjustment: Setting `force` and `Speedjump` helps maintain consistent physics behavior in the game, ensuring smooth and realistic movement for Bird.
* Finally, this enforces rules related to platform collisions, which are essential for platformer-style games where accurate collision detection and response are crucial to the gameplay

How they link:

Usability features:

* Described, justifying choices made, the usability features to be included in the solution.

Controls:



Move right

Jump

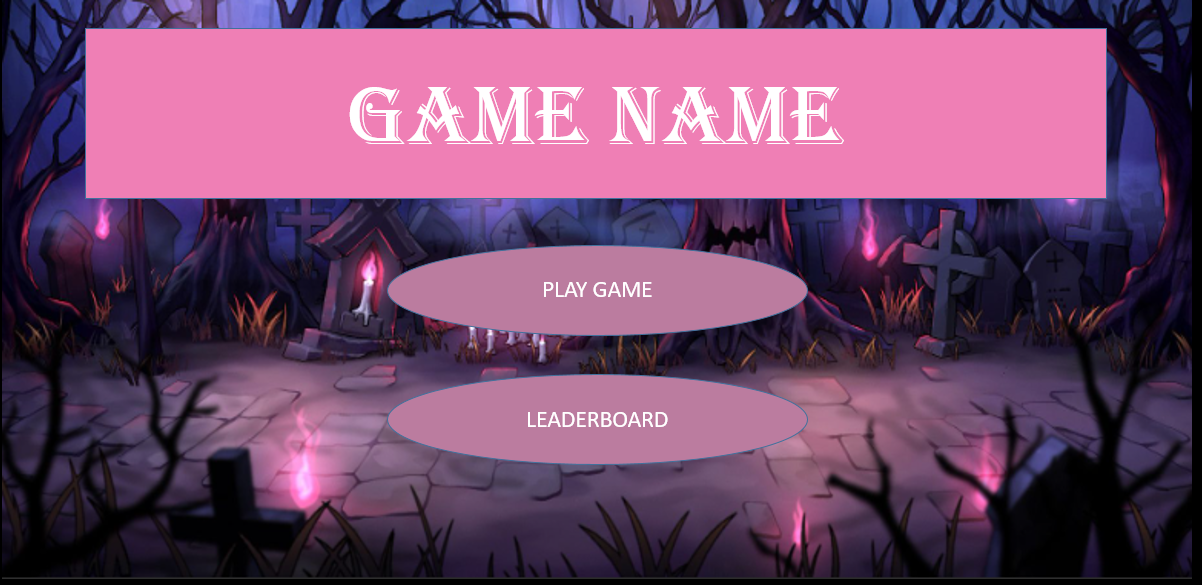
Move left

Above is the layout of the control scheme tailored for seamless player interaction in the game. I have chosen these keys to make playing easy and natural.

By using the A and D keys from the WASD keys, I have thoughtfully designed the control scheme to ensure that I provide an intuitive experience for the players. The A and D keys allows players who are familiar with gaming to effortlessly navigate the character. Furthermore, for players who are new to gaming, the proximity between the A and D keys makes it easier for the players to identify and use them.

In addition, using the space bar to jump allows the players who are new to gaming to easily locate it due to its prominent size and press without much difficulty, thus achieving a 100% success rate.

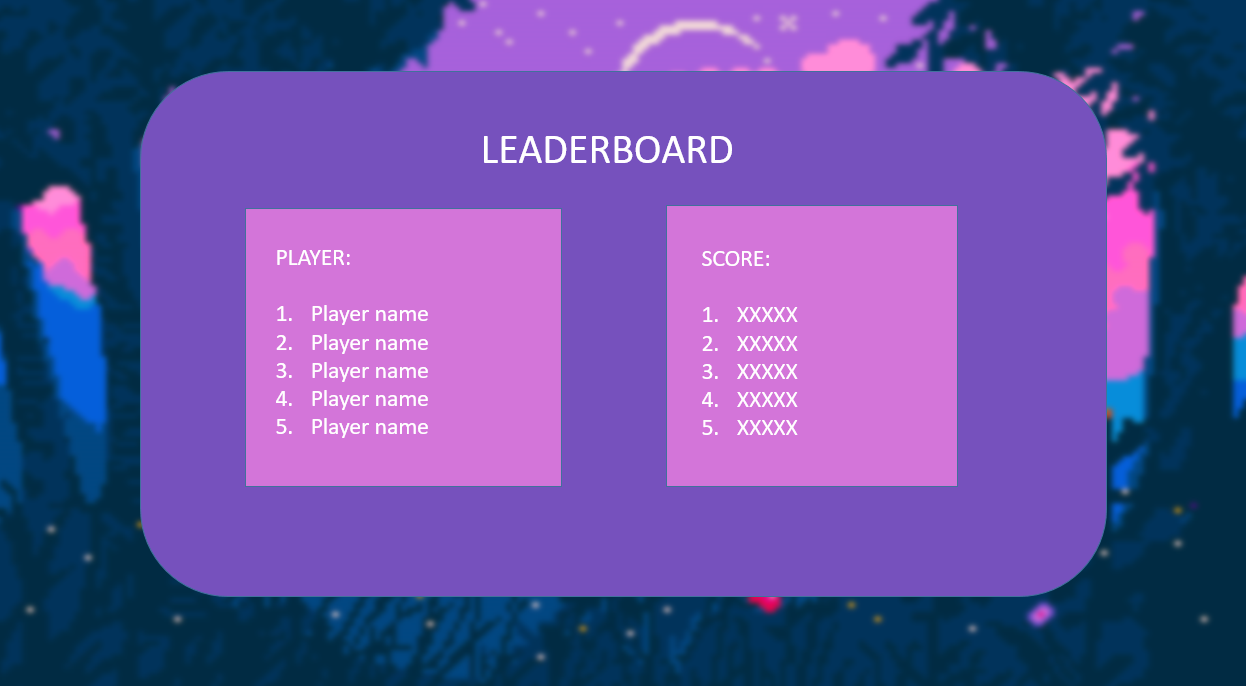
Menu:



This is how my menu will look with other things added to it of course.

The menu layout follows a familiar pattern seen in games over the past decade. The game's title is prominently displayed at the top, with selectable buttons arranged below it. This design is intuitive for both experienced gamers and newcomers alike. The title naturally draws attention, guiding the viewer's gaze to the selectable buttons. With a clear left-to-right flow, navigating the menu becomes effortless for anyone, ensuring accessibility for all users.

Leaderboard:



The leader board showcases the top 5 scores, arranged from highest to lowest. Clear, easy-to-read fonts are used for better readability. The background colour is a lighter shade of purple, complemented by white text for enhanced contrast, ensuring all users are able to read it. To exit the game, a noticeable red button is placed at the bottom centre. Its size and colour help users locate it easily and understand its purpose.

Game interface:

Text:

# 

# **Data Structures:**

* Identified and justified the key variables / data structures / classes (as appropriate to the proposed solution) justifying and explaining any necessary validation.

Key variables

All the variables listed below are private.

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Variable Name | Data type | Justification |
| Player: | Bird | PictureBox |  |
|  | Go\_left | Boolean |  |
|  | Go\_Right | Boolean |  |
|  | jump | Boolean |  |
|  | Speedjump | Integer |  |
|  | SpeedBird | Integer |  |
|  | force | Integer |  |
| Player lives: | lives | Integer |  |
| Enemy: | enemy | PictureBox |  |
|  | enemyAlive | Boolean |  |

Validation:

In my game, validation isn't a big deal because the user inputs are already limited since it's a game and they don’t really need to input text, so that doesn’t need validation. For the game controls, only the right keys will work, so I just need to validate those and I will also need to validate the mouse clicks to make sure the player is clicking within the bounds of the game. Furthermore, when players enter their usernames for the leader board, I need to make sure they're between 1 and 15 characters. I want to keep things simple and make sure everything runs smoothly.

TEST DATA:

Menu:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Number | Text | Input | Validity | Expected Outcome | Justification |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Player:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Number | Text | Input | Validity | Expected Outcome | Justification |
|  | Right Movement | D key | Valid | Player moves right | This shows that the player can move right |
|  | Left Movement | A key | Valid | Player moves left | This shows that the player can move left |
|  | Jumping | Space Bar | Valid | Player jumps up and then comes back down | This shows that the player can jump |
|  | Pillar Collision | None | N/A | Player stops moving left or right | This shows that the player can’t go through pillars |
|  | Floor Collision | None | N/A | Player stops falling | This shows that the player can’t go through the floor |
|  | Collecting a coin | None | N/A | The coin disappears and added to the coin count at the top right corner of the screen | This shows that the player can collect coins |
|  | Getting hurt by the enemy | None | N/A | Player loses some lives | This shows that the player can be hurt |
|  | Getting killed by the enemy | None | N/A | Player dies | This shows that the player can die |
|  | Invalid inputs | Anything other than the ‘A’ and ‘D’ keys or space bar | Invalid | No change | This shows that the player can’t be affected by invalid inputs |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Leader board:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Number | Text | Input | Validity | Expected Outcome | Justification |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Coin:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Number | Text | Input | Validity | Expected Outcome | Justification |
|  | Being collected | None | N/A | Coin disappears | Shows that the player collected the coin |

Enemy:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Number | Text | Input | Validity | Expected Outcome | Justification |
|  | Hurting the player | None | N/A | The player is hurt | This shows that the enemy can hurt the player |
|  | Getting killed by the Player | None | N/A | The enemy dies | This shows that the enemy can die |
|  | Pillar Collision | None | N/A | The enemy stops moving left or right | This shows that the enemy can’t go through the pillars |
|  | Floor Collision | None | N/A | The enemy stops falling | This shows that the enemy can’t go through the floor |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# **Further Post Development Data**

* Identified and justified and further data to be used in post development phase.

In the post-development phase, the Leader Board will necessitate some initial data population. This is essential as the Leader Board's functionality hinges on having a set of scores to organize and exhibit.

Beyond the Leader Board, the project won't necessitate additional data.

## Development Of Coded Solution

# **DEVELOPMENT AND TESTING:**

* Provided evidence of each stage of the iterative development process for a coded solution relating this to the break down of the problem from the analysis stage and explaining what they did and justifying why.

STAGE 1: THE MENU

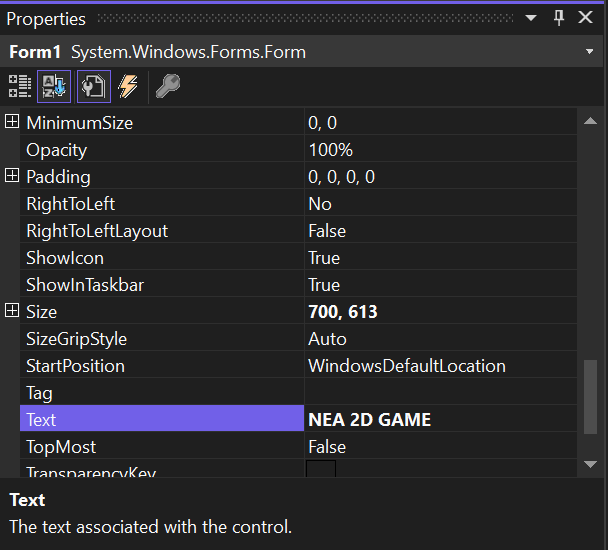
Sprite class:

Button class:

STAGE 2: THE GAME SCREEN

Making the Game Screen:

|  |  |
| --- | --- |
| Image | Justification |
|  | This is the background image I am using for my game, this is a large image with a width of 1920 pixels and a height of 1125 pixels. Using this as my background, I will be able to scroll this image left to right. By using this pixelated background, I can recreate the visual style of classic games such as ‘Super Mario Bros’ which is appealing to my stakeholder and also other gamers who enjoying playing 2D games with pixelated aesthetics. |
|  | This is the coin image I will be using for my game, and it is an animated coin GIF. I chose an animated coin GIF rather than a static one since games for younger kids such as my stakeholder should prioritise fun and enjoyment. This animated element will allow me to add an element of fun and whimsy feel to the game, making the experience more entertaining for the younger kids similar to my stakeholder. |
|  | This is the platform image, all the platforms in my game will have this as their background image. By using the same platform image throughout my game, it will allow me to ensure visual consistency so players can easily recognise and understand the platforms. |
|  | This is the closed portal image that will be at the end of my game that will change to an opened portal will be fun for younger kids such as my stakeholder since it provides a sense of achievement and imagination. This adds depth and excitement to the game experience which will make it more enjoyable for children. |
|  | This is the treasure image I will be using in my game. The treasures found in the treasure box can serve as collectibles or unlockable items that enhance the gameplay. |
|  | This is the open portal image, when the player collects the key and collides with the closed portal image it will change to this image |
|  | This is my character I created and will a ‘Bird’. This is how it looks when it’s moving right using the ‘D; key. |
|  | This is the Bird sprite and this is how it looks when it’s moving left using ‘A’ key. In the real world, people or objects typically face the direction they’re moving. By having the character face the direction of its movement, it adds a level of realism to the game and helps immerse my stakeholder/ other players in the game world. |
|  |  |
|  |  |
|  |  |



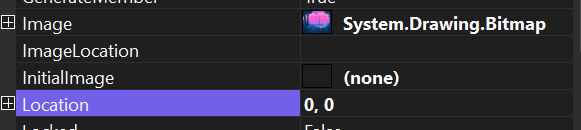
The screenshot above shows that I changed the ‘Form1’ text into ‘NEA 2D GAME’ since that is my title. I also made the size 700,613 since that is how it will appear to the players.



From the toolbox, I dragged and dropped this picture box into the form. This will be used as the main background for my game.

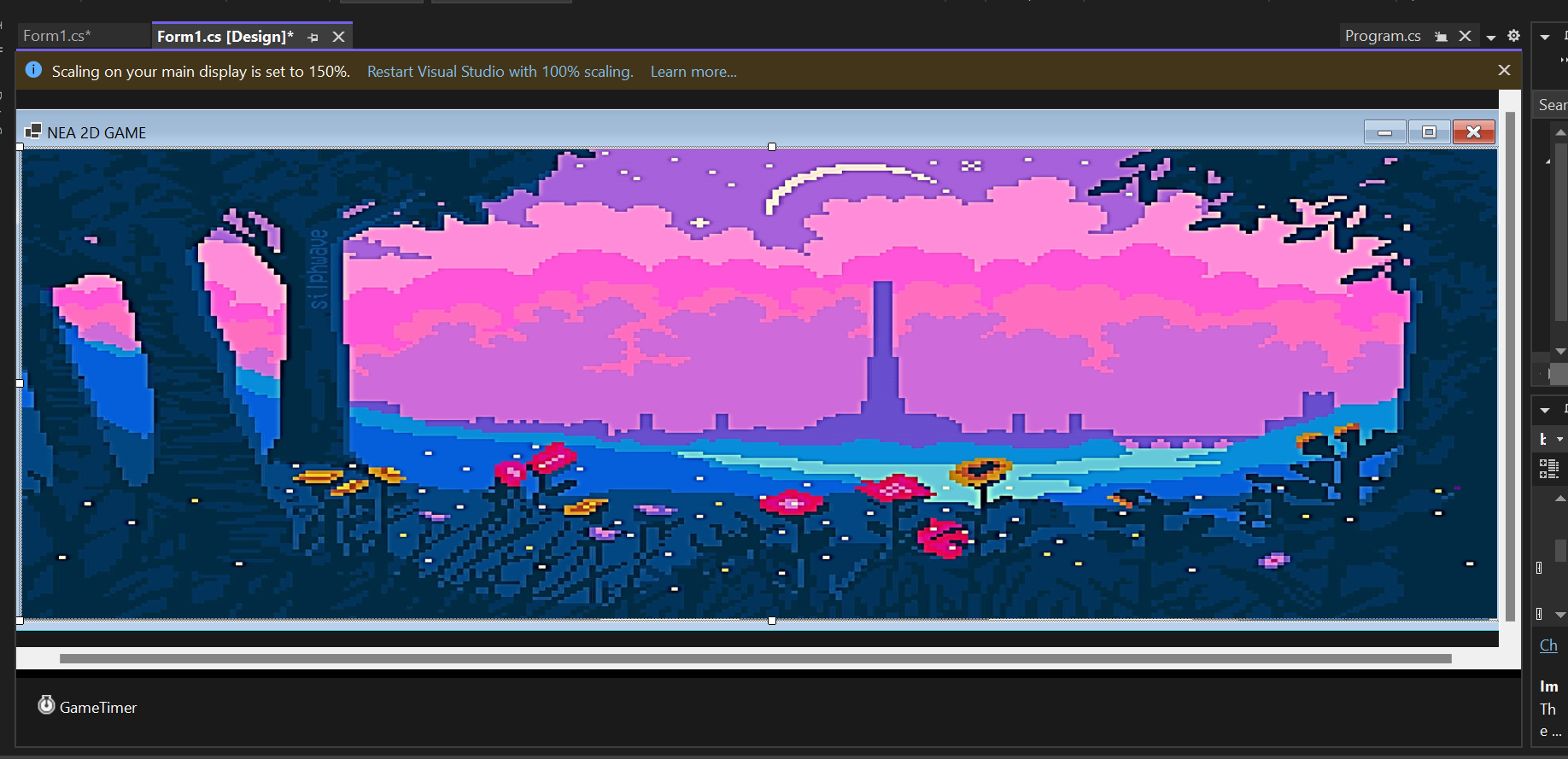
Once I have done that I changed the following in its properties window:





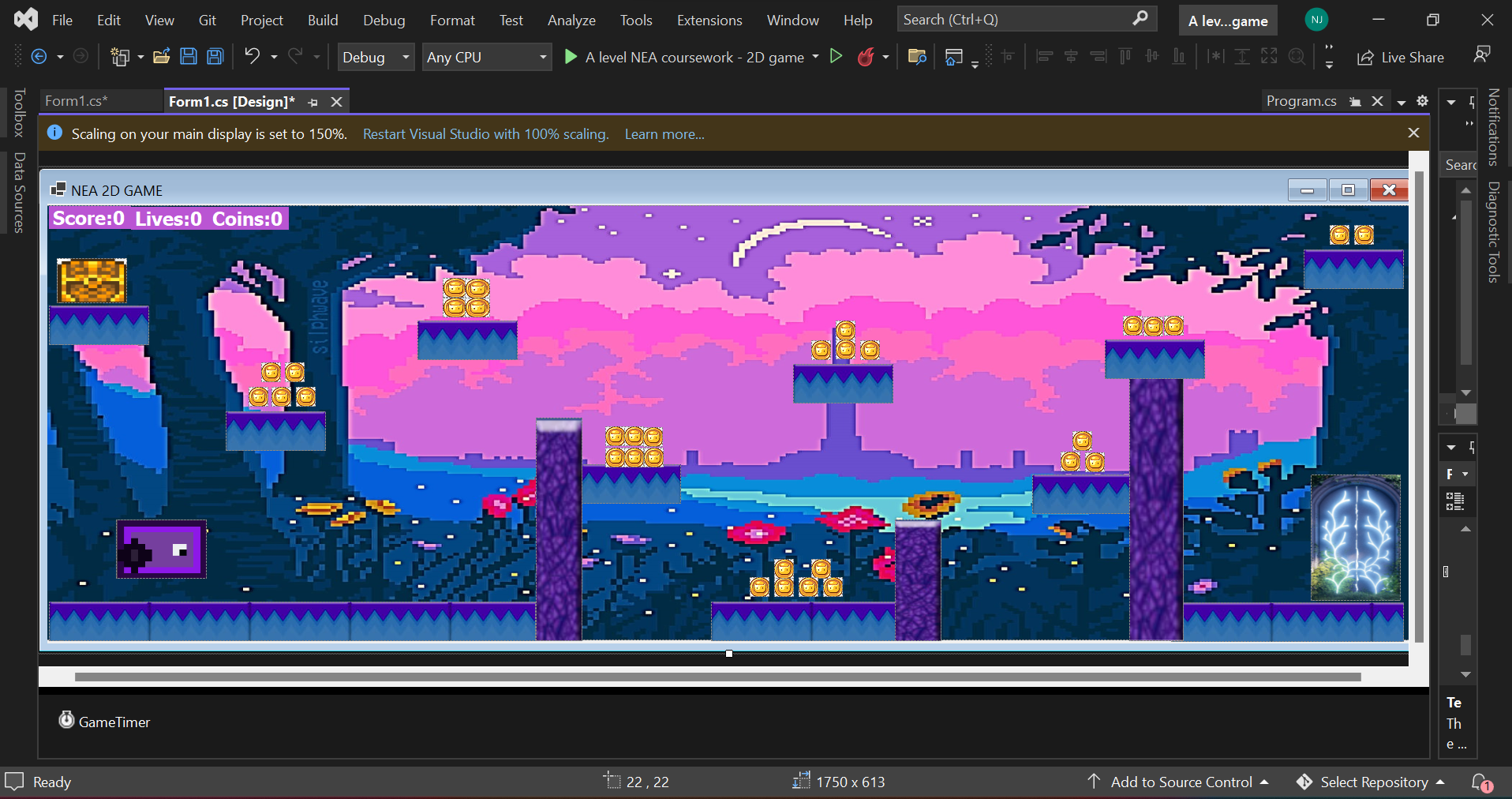
Changed the location to 0,0 and then added the background image I showed earlier.

BEFORE PUTTING ANY OTHER IMAGES:



This is how the screen looks like on the display now that I put the background image in. I had to stretch it out so that I can fully fit all my other images and also make it a side scrolling game.

AFTER:



# **Evidence Of Prototype Versions**

* Provided evidence of prototype versions of their solution for each stage of the process.

# **Modular & Well-Structured Solution**

* The solution will be well structured and modular in nature.

# **Annotations**

* Code will be annotated to aid future maintenance of the system.

# **Variables and structures**

* All variables and structures will be appropriately named.