AH Computing Science Project Proposal

Contents

Idea

This project will be of a game called Netris. This game is about stacking blocks to create a full horizontal line and get points, the game gets harder as you continue as the blocks move faster towards the bottom of the window, giving you less time to make a decision on how to rotate and place the block. You lose when the blocks make it to the top of the screen and there's no more space. This game will be modelled to be similar to the original but some things may be taken out to avoid copyright.

Target Audience

- Main Age & Gender Range: est. 5+

- Interests: Games/Retro games
- Other:
    - You must own a computer to play this game
    - You must be semi proficient with computers in order to install python due to the game being programmed in the language

Requirements

Input Validation:

- The player can use the up key to rotate the blocks round, the left and right arrow keys to move the blocks left and right.
- The scoreboard will only allow alphabetical characters, the enter and backspace key and a maximum of 3 characters should be inputted
- Mouse input is required for the menus, of which only the left click is validated
- Files will be available for the program to take as parameters and modify the values, the values should also be validated

Reading & Modifying Stored Data:

- This program will only store the top 5 scores into a text file which will be used to construct a scoreboard. The top 5 scores will also be saved into the same file again. A text file will also be used to store the state of settings

Sorting Algorithms:

- A standard algorithm will be used to sort the high scores before they are written to a file.

An Array of Objects:

- An array of objects will be used to store the score data & The GUI elements whilst the data is being manipulated using code.

Research

Feasibility Review

Technical Study:

- This game is required to be object oriented to cut down on implementation time and to meet some requirements set by the SQA. The Python programming language is used for this purpose due to it already being object oriented, easy to obtain and install, implementation time will be cut down due to me already being proficient in the language as well.

- This game will require a GUI, for this I have chosen to use the python library, Pygame. This requires Pip (A python packaging installer). This is feasible due to Pip coming with any stable release of Python and installing Pygame is easy due to it being a one line command in the terminal.

- In order to fulfil one of the requirements the SQA has given, I will be using pyodbc for my SQL query to a database. This is feasible due to the installation only taking a one line command in the terminal.

- I will require documentation for how to use the Pygame library efficiently. This is feasible due to many websites detailing the many methods and classes that are provided with Pygame. The documentation with which I will be referencing will be in the Pygame website.

Economic Study:

- This project has no cost attached to the development of this program. All the software used is free and the purchase of hardware is not required, a licence is also not required in the creation of this project.

Legal Study:

Data Protection:

- Data Being Collected:
    - Highscores
    - Player Names (3 Characters Maximum)

- Copyright
    - The copyright for the original depiction of this game is held by The Tetris Company, This could be a problem. I will need to use resources that are publicly available in order to prevent copyright infringement.

    - But due to this project not being a commercial product, this is not applicable to myself as there is no monetary gain involved, however, I will still follow these guidelines to the best of my ability.

- Schedule:
    - This project is complex enough to be created within the timeframe allowed, including holidays and unforeseen events such as illness.

Sample Survey For Analysis

Netris End User Survey

Age & Gender: _____

Have you heard of Tetris Before	Yes	No	Don't Know
Do you know how to play Tetris	Yes	No	Don't Know
Do you Often Play Video games	Yes	No	Don't Know
Would you Expect Tetris to have a highscore table	Yes	No	Don't Know
If Yes, how many scores would be displayed	5	10	Other _____
Would you expect Tetris to have background music	Yes	No	Don't Know
What age range would be fitting for this game	5	10	Other _____

How many devices do you own

_____

What type of devices do you own

_____

What is the primary device you use

_____

What device would you play this game on

_____

Is there anything else you would like to see in Netris?

Survey Results Given in Appendix A For Further Reading

Analysis

Analysis of the Survey Results

Video Games

- **Knowledge of Tetris**: 100% of the respondents knew about Tetris and how to play it (rules etc).

- **Playing Video Games**: ~93% of the respondents play video games often whilst the other ~7% do not. There was no discernible difference between the respondents gender and how much they played video games, however, since the majority of the respondents were male, this tips the scales slightly with only one female respondent who does not play video games often.

In conclusion, Since the majority of respondents were male the target audience will be male, however, that does not exclude any other gender from the end user group due to the minor difference in who did and didn't play video games often and the pool of respondents.

Game Features

- **Highscore Table Expected**: All respondents expect Tetris to have a highscore table therefore one will be implemented

- **High score Table**: ~53% of the respondents wanted 10 scores to be displayed on the high score board with ~20% wanting 5 and the rest of the respondents picking another number

- **Background Music**: ~93% of respondents expected Tetris to have background music, therefore I shall add some background music to the game that is copyright free

- **Age Range**: The majority of respondents felt that any age range would fit Tetris, therefore the target age range will be all inclusive.

In conclusion, all respondents expected a high score table for Tetris so one will be implemented with a maximum number of scores being 10 since that was the majority voted upon. I will also add background music because the majority of

respondents expected there to be some sort of music, this will be within the public domain to make sure that there is no copyright infringement.

End User Device Ownership

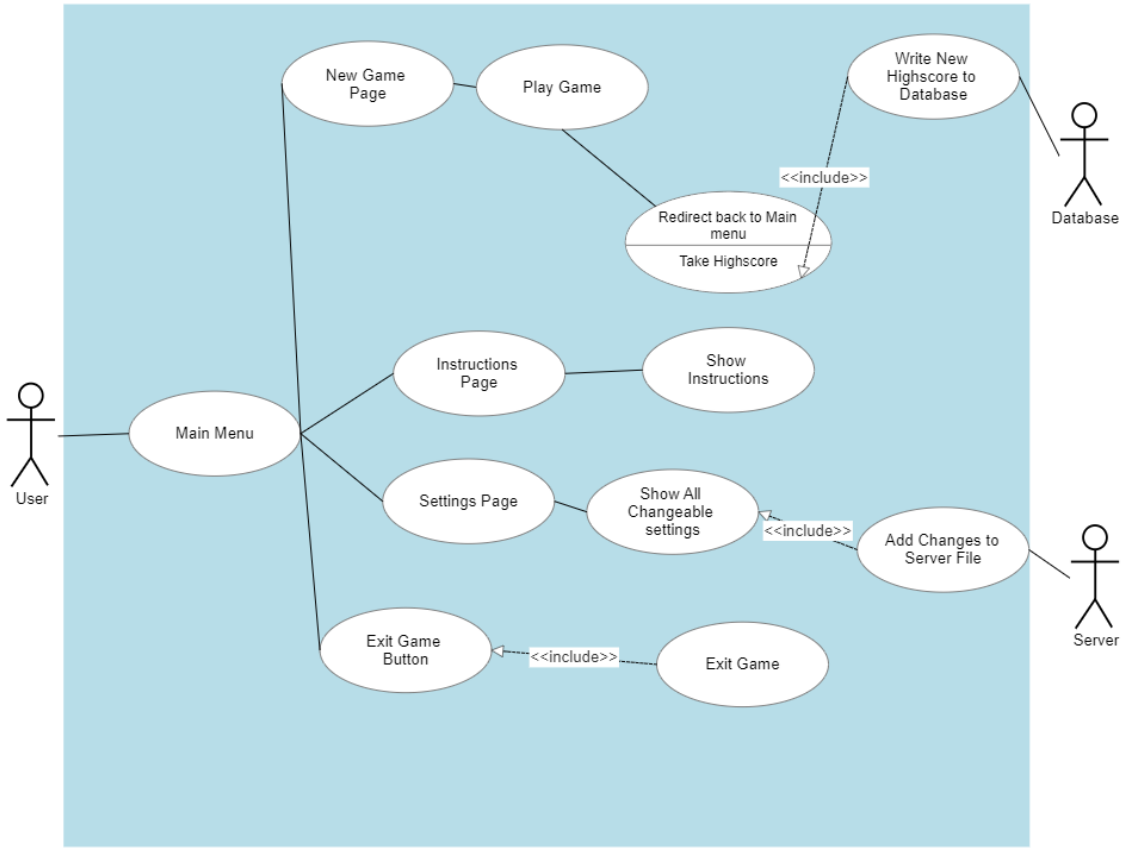
- **How many devices do you own**: The majority of respondents owned between 2 - 5 devices

- **Primary Device**: The majority of respondents mainly use their phones with the 2nd most common device being a PC or laptop

- **What device would you play this on:** The majority of respondents would play this game on their phone, however, due to time constraints I will be making this game for Windows PC exclusively because that is the easiest platform to develop for within the time constraints of this project.

In conclusion, most respondents own between 2 - 5 devices so multi platform support would be beneficial, however due to time limitations I will not be able to support more than 1 platform (Windows PC) because that would require me learning many, platform specific, languages that I do not have the time to do.

UML Case Diagram



Requirements Specification

End Users

- The end users will be any age who enjoy playing video games
- Users must have basic knowledge of how to use a computer & they must own a UK qwerty keyboard.

End User Requirements

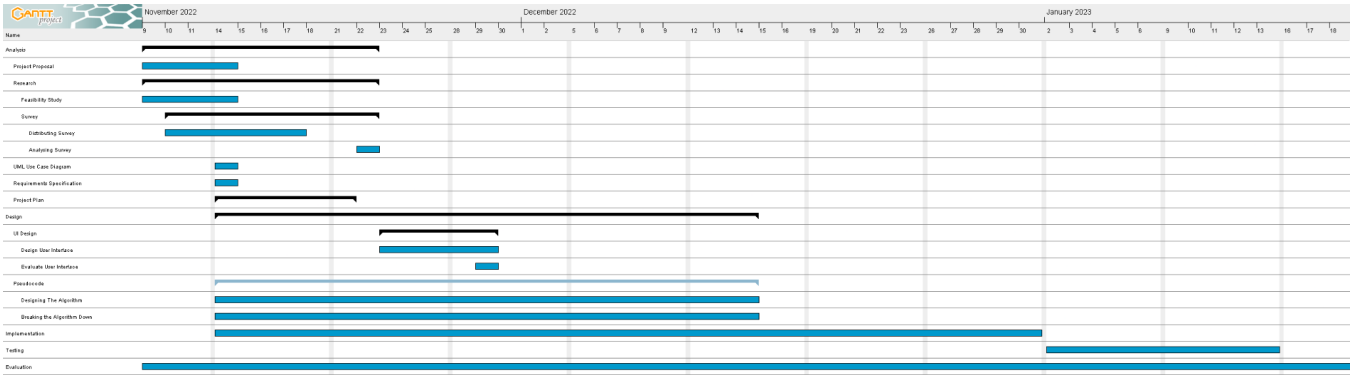
- The user must be able to play the game without the window lagging or stuttering during gameplay
- The user must be able to play the game easily, meaning the game must be easy enough to keep up with

Functional Requirements

- There must be rules/instructions page to help the user understand how to play the game if they don't know how to already
- The user must own a modern windows computer in order to play this game
- The user must have a keyboard, mouse and some kind of screen in order to interact with the game and see the game window
- The user must have python, pygame and pyodbc installed to run this game properly

Project Plan

Gantt Chart



(Note: Full Image can be found in Appendix A: Analysis)

Resources

- Pen/Pencil & Paper for design prototyping
- SQA Project Specification & Any other appropriate SQA documents
- Computer with mouse, monitor & keyboard
- Computer Software & Hardware
    - Gantt Project (Gantt Chart Creation)
    - VS Code (Implementation)
    - Python & Libraries (Pygame, Pip, Pyodbc)
    - Github (For hosting files during development & timeline of files)
    - Sufficient Storage Space
    - Balsamiq (For design of UI)

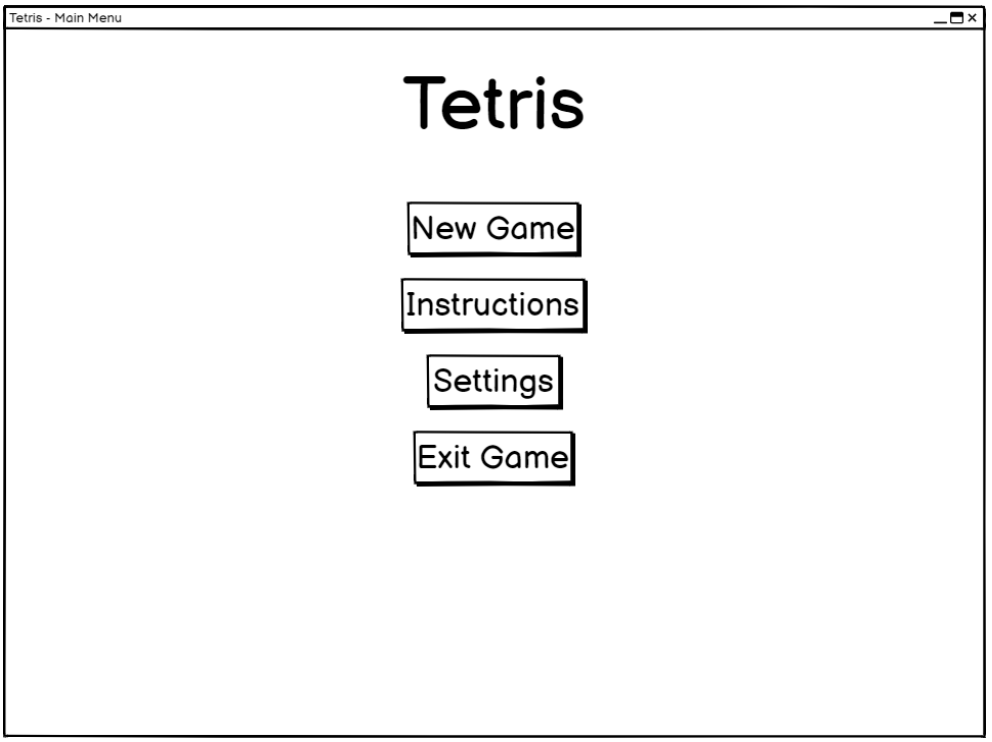
- Audio
    - Sound Effects: https://drpetter.se/project_sfxr.html

- Websites Used
    - Pygame Documentation Website https://www.pygame.org/docs
    - Google Docs (For Write Up)
    - Stack Overflow & other coding help forums

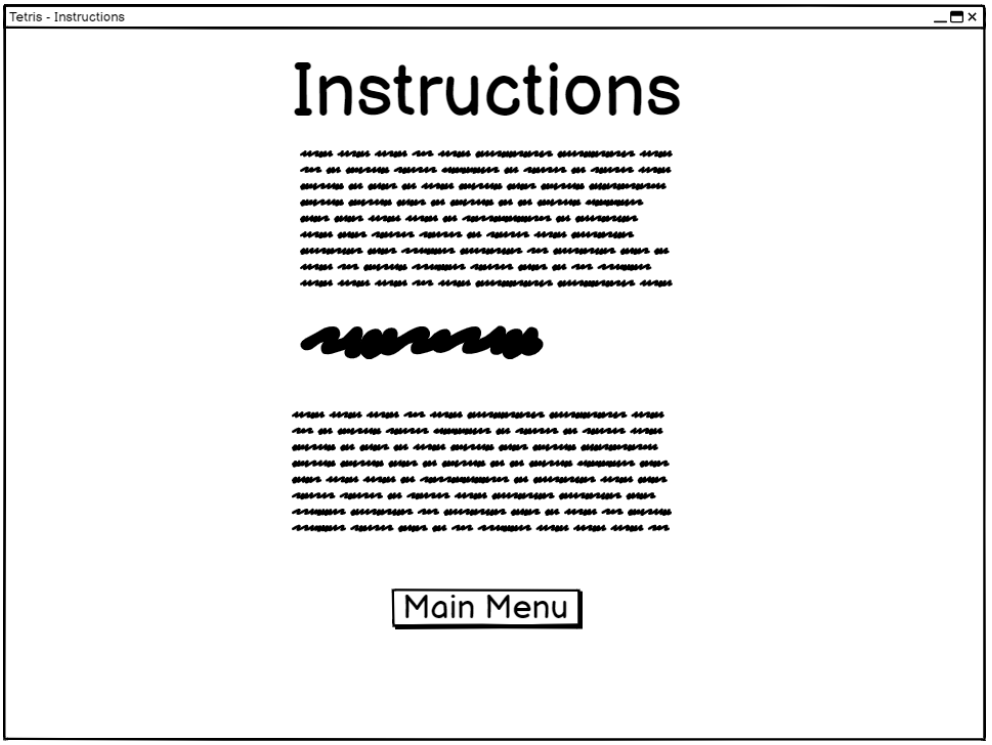
Design

UI Wireframes

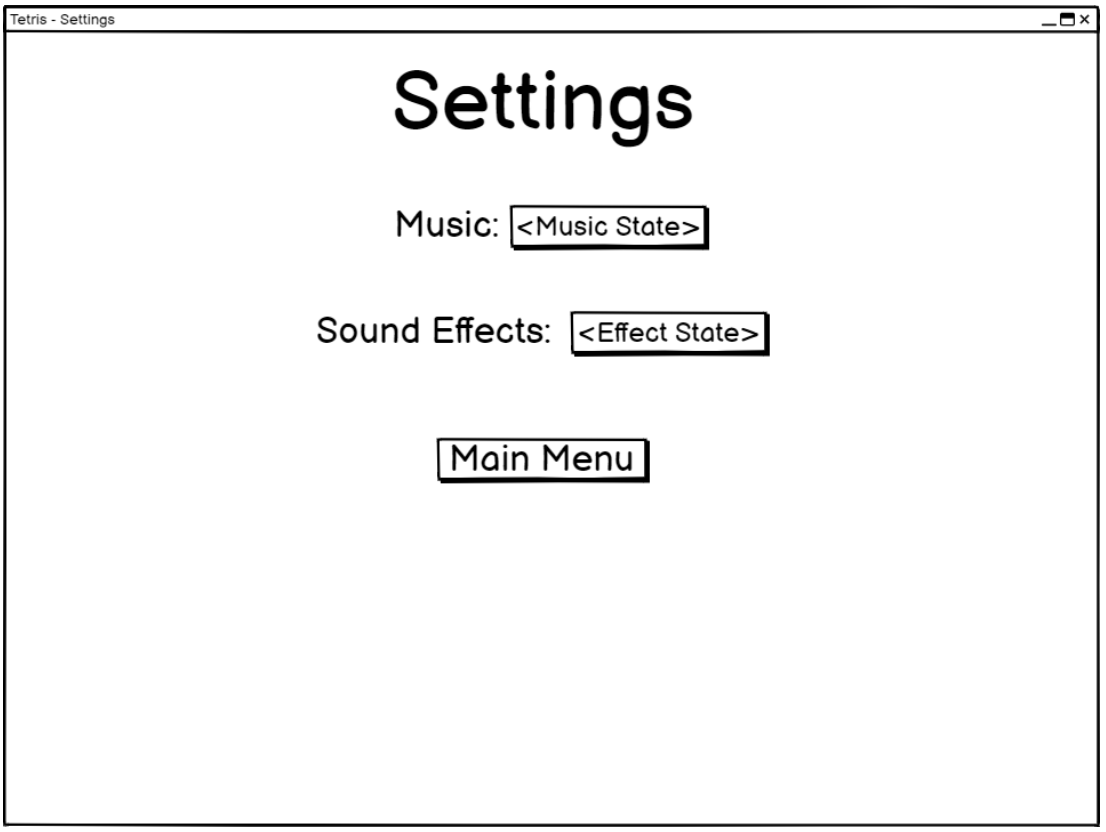
Home



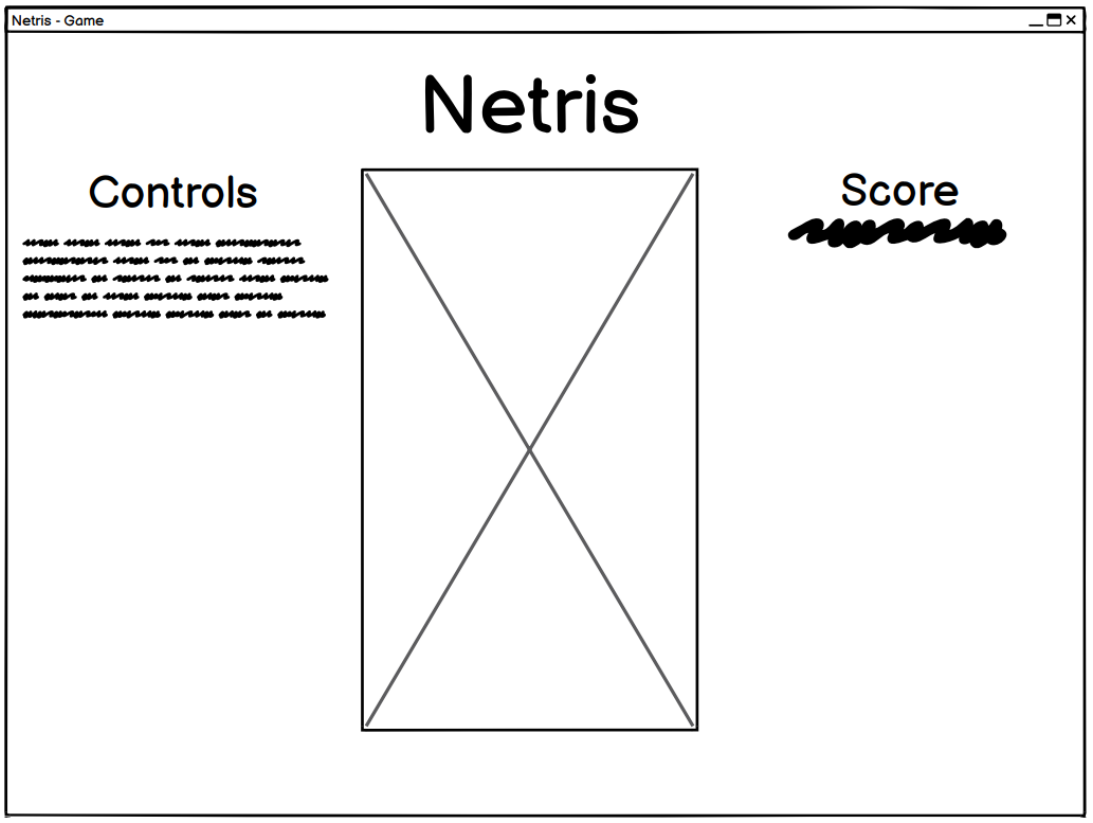
Instructions



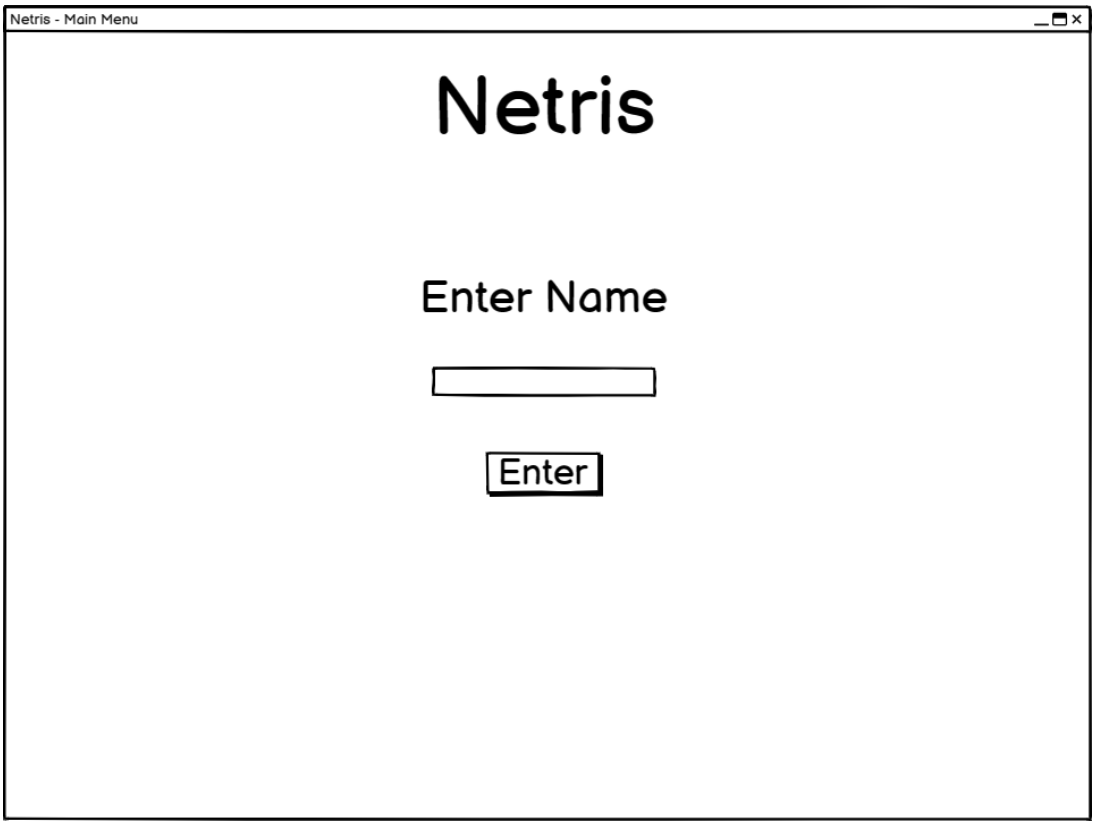
Settings



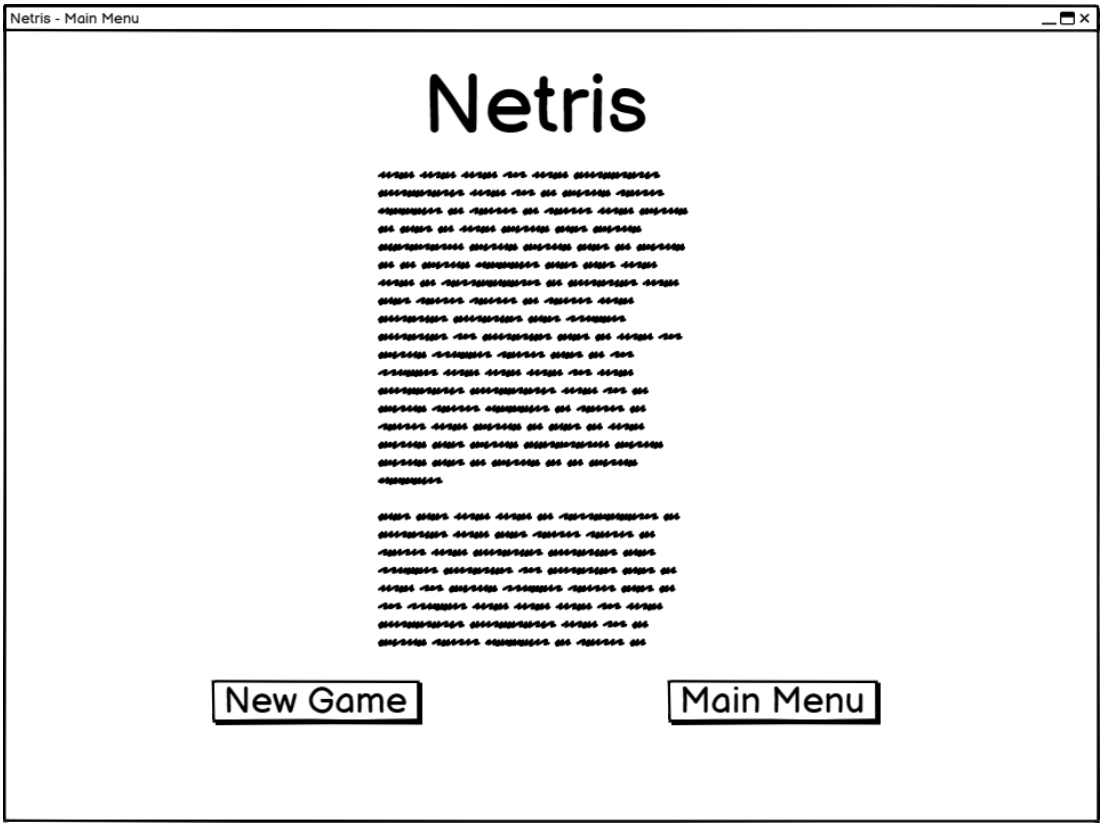
Game



Name Input



Highscore Table



(Note: All image files can be found in Appendix B: Design)

Pseudocode

Software Design & Development

Standard Algorithm

```
1.1 FUNCTION BubbleSortScores(THIS, LIST scoresList, BOOLEAN dev INITIALLY TRUE) RETURNS LIST OF LISTS
1.2     IF THIS.name != 'PLA' AND THIS.score != 0 THEN: scoresList.append(Highscore(THIS.name, THIS.score))
END IF
1.3
1.4     FOR i IN RANGE(LENGTH scoresList) DO
1.5         FOR j IN RANGE(LENGTH scoresList - i - 1) DO
1.6             IF scoresList[j+1].score > scoresList[j].score THEN
1.7                 SET scoresList[j], scoresList[j+1] TO scoresList[j+1], scoresList[j]
1.8             END IF
1.9         END FOR
1.10    END FOR
1.11 RETURN [[score.name, score.score] FOR EACH score IN scoreList[:5] END FOR]
        IF NOT dev ELSE [[score.name, score.score] FOR EACH score IN scoreList END FOR] END IF
1.12 END FUNCTION
```

Array of Objects

```
1.1 DECLARE GUIObjects INITIALLY AS ARRAY of OBJECT [
1.2         Text([480, 90], 'Netris', 106),
1.3         Btn('New Game', [480, 310], 230, 41, 32),
1.4         Btn('Instructions', [480, 360], 300, 41, 32),
1.5         Btn('Settings', [480, 410], 200, 41, 32),
1.6         Btn('Developer Mode', [480, 600], 210, 41, 16),
1.7         Btn('Exit Game', [480, 460], 235, 41, 32)
1.8 ]
```

(NOTE: More than one array of objects were used within this project, to view every one of them see Appendix B: Design for further reading)

Database Design & Development

Open/Close Database Connection to Execute SQL Query

```
1.1 <@staticmethod>
1.2 FUNCTION CommitToDb(LIST scores) RETURNS LIST OF TUPLES
1.3         DECLARE topScores INITIALLY []
1.4         SET scores TO scores[0]
1.5
1.6         TRY
1.7                 DECLARE conn_str INITIALLY f'''DRIVER=SQL SERVER;
                                                   SERVER={SERVER_NAME};
                                                   DATABASE=highscores;
                                                   Trust_Connection=yes;'''
1.8
1.9                 DECLARE conn INITIALLY dbc.connect(conn_str) # open database connection
1.10                DECLARE cursor INITIALLY conn.cursor()
1.11                cursor.execute('TRUNCATE TABLE highscore;')
```

```
1.12
1.13            DECLARE ins_query INITIALLY f'"INSERT INTO highscore (name, score) VALUES (?, ?);"'
1.14            FOR EACH name, score IN scores DO cursor.execute(ins_query, (name, score)) END FOR
1.15
1.16            conn.commit()
1.17            Color.prints('Committed Data to Database Successfully')
1.18
1.19            TRY
1.20                    DECLARE data INITIALLY cursor.execute('SELECT * FROM highscore;')
1.21
1.22                    SET topScores TO [(name, score) FOR EACH name, score IN data END FOR]
1.23                    Color.prints('Successfully retrieved score data from database')
1.24                    conn.close() # close database connection
1.25
1.26                    Color.prints(f'Data: {topScores}')
1.27                    RETURN topScores
1.28            EXCEPT Exception as e
1.29                    Color.printe(f'Error whilst trying to retrieve score data\n{e}')
1.30            END TRY
1.30    EXCEPT Exception as e
1.31            Color.printe(f'Error: There was an unexpected error whilst trying to commit data to the sql
database\n{e}')
1.32    END TRY
1.33 END FUNCTION
```

(Note: Pseudocode is part of Appendix B: Design for further reading)

Database Design

Data Dictionary

Field Name	Constrain	Data Type	Field Size	Description
id	Primary Key	Integer	variable	Auto Generated
name	Not Null	Text	3	Users Name
score	Not Null	Integer	variable	Users Score

Structure of Table

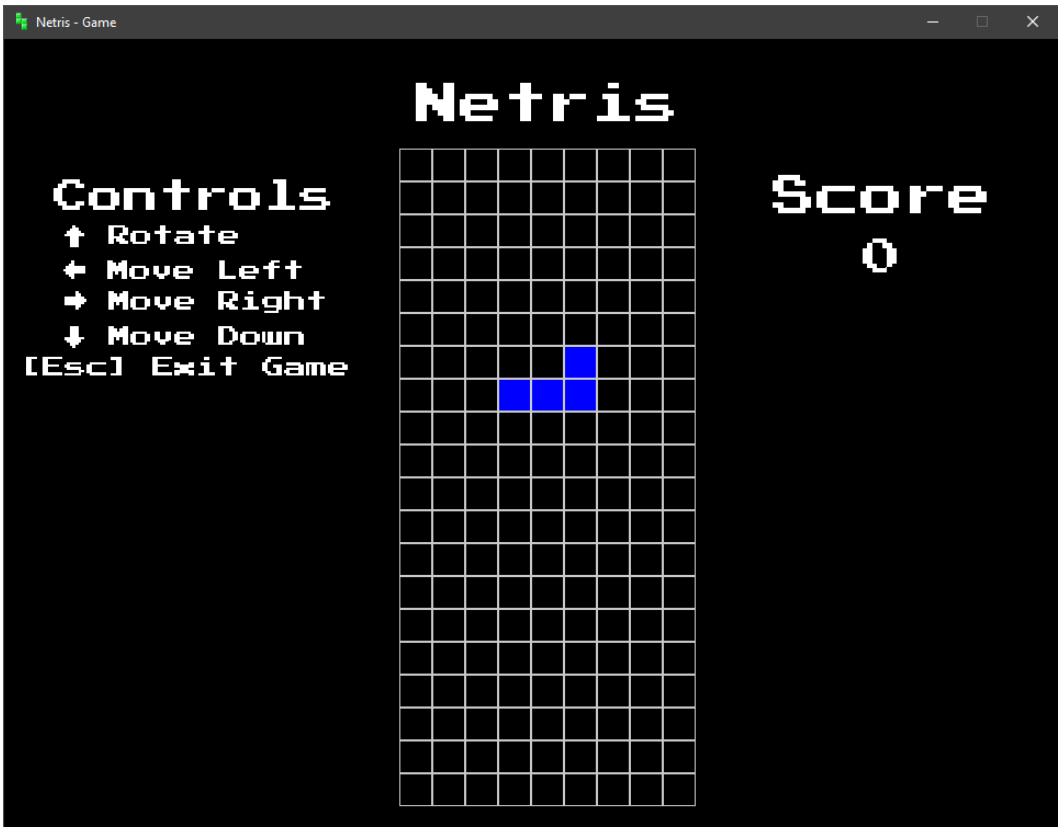
```
CREATE TABLE highscore {
        id int NOT NULL AUTO_INCREMENT,
        name varchar(3) NOT NULL,
        score int NOT NULL,
        PRIMARY KEY (id)
};
```

Implementation

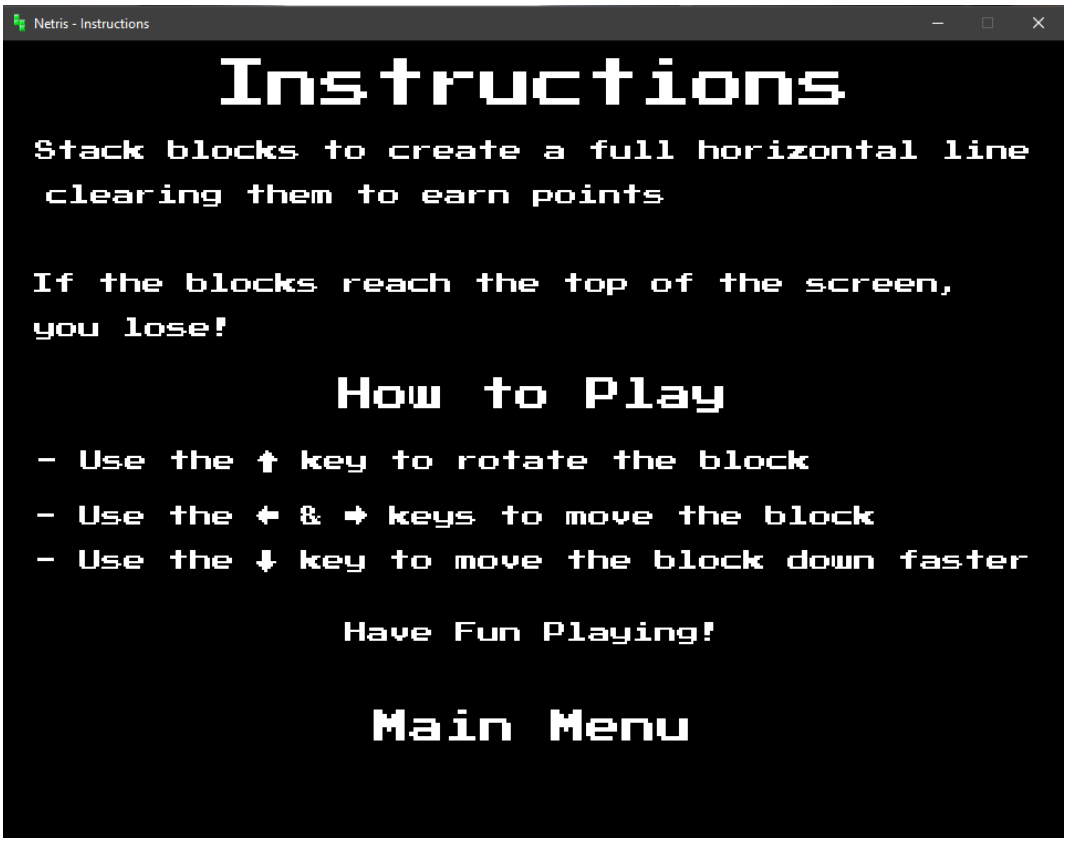
UI Design

Main Menu

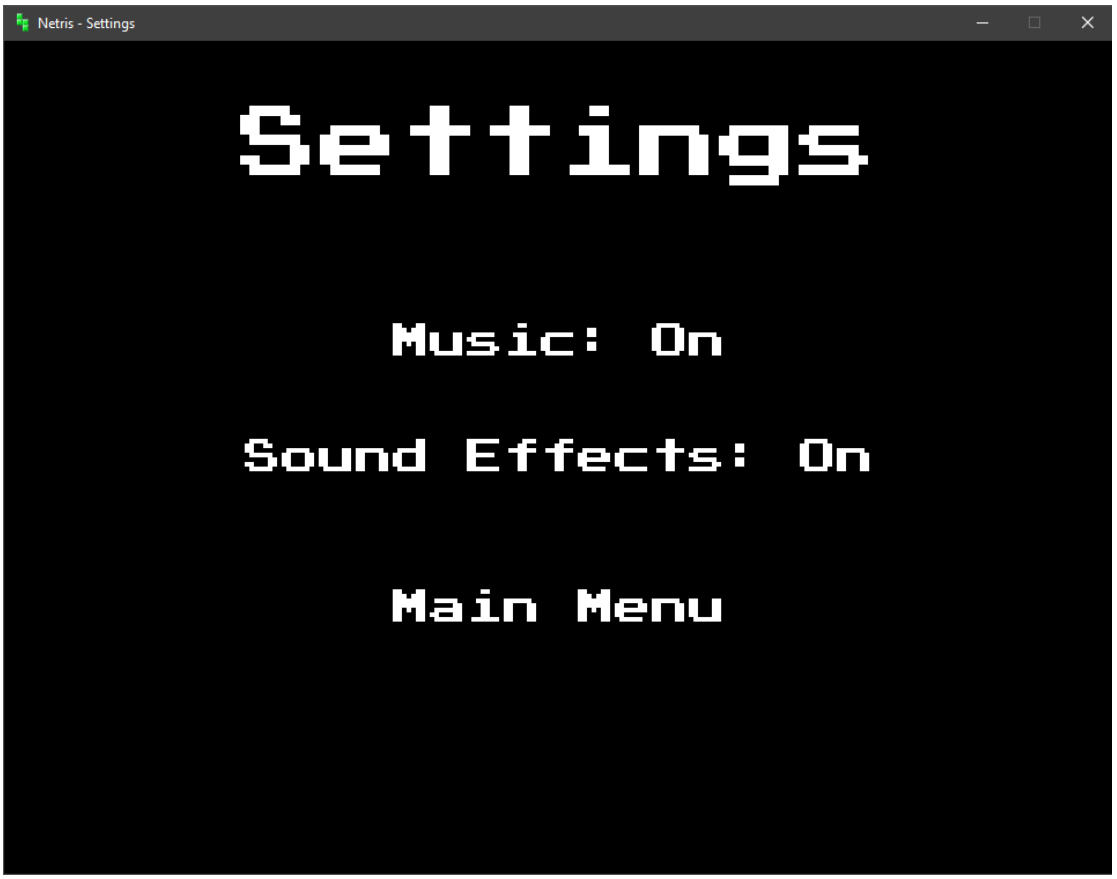
New Game



Instructions



Settings


Netris - Settings
Settings
Music: On
Sound Effects: On
Main Menu

Score Input

Highscores

Code

Standard Algorithm

```
def BubbleSortScores(self, scoreList:list, dev:bool=False) -> list[list]:
	'''Sorts Scores in Order of Highest First, Lowest Last

	Args:
	    -	scoreList (list): List of Highscore Objects

	Returns:
	    -	list[list]: List of lists in the form of [name, score]
	'''

	if self.name != 'PLA' and self.score != 0:
		scoreList.append(Highscore(self.name, self.score))

	for i in range(len(scoreList)):
		for j in range (len(scoreList)-i-1):
			if scoreList[j+1].score > scoreList[j].score:
				scoreList[j], scoreList[j+1] = scoreList[j+1], scoreList[j]

	return [[score.name, score.score] for score in scoreList[:5]] if not dev else [[score.name, score.score]
	for score in scoreList] # list comprehension
```

Array of Objects

```
# List of GUI Objects
GUIObjects = [
	Text([480, 90], 'Netris', 106), # Game Title
	Btn('New Game', [480, 310], 260, 41, 32), # New Game Button
	Btn('Instructions', [480, 360], 400, 41, 32), # Instructions Button
	Btn('Settings', [480, 410], 260, 41, 32), # Settings Button
	Btn('Developer Mode', [480, 600], 210, 41, 16),
	Btn('Exit Game', [480, 460], 300, 41, 32) # Exit Button
]
```

(Note: All code can be found in Appendix C: Implementation for further reading)

Testing

Test Plan

Home Page

No Normal, Exceptional or Extreme data for this page (due to mouse input only)

UI:

- Cursor Visible

- Black Background Colour
- White Text Colour
- UI in correct positions
- Correct Text Usage (e.g title is the right text)
- Correct font & font size used

Usability:
- Ease of Use & Ease of Understanding
- Functionality made clear for each UI object
- Readability

Functionality
- Do buttons redirect using their functions correctly
- Do buttons turn red on hover
- Do buttons play a sound when hovered & clicked if the settings are set
- Does the escape key quit the program

New Game Page

Normal Input: Up, Down, Left, Right Arrows, Escape, score file
Exceptional Input: Multiple combinations of Arrow Keys, empty score file
Extreme Input: Alphanumeric keys, no score file

UI:
- Cursor Invisible
- White Text Colour
- Black Background
- Grid in correct position & size
- UI in correct positions, sizes & font
- Display updatable score on right side of screen
- Display controls on left side of screen
- Draw blocks at top of screen with correct colour & orientation

Usability:
- Ease of Use & Ease of Understanding
- Readability

Functionality:
- If music setting is True then play background music on repeat
- If sound effect setting is True then play sound effects
- If Up arrow pressed rotate block
- If left or right allow is pressed move block
- If down arrow is pressed move block down

- Don’t move or rotate block if at grid boundary positions
- If block reached grid base then stop moving, add to score then add block to group then draw new block
- If block collides with block in group stop moving block, add to score then draw new block
- If block group reaches top of grid redirect to name input page
- If blocks fill entire row, add to score then remove row of blocks and move group down

Instructions Page

No Normal, Exceptional or Extreme data for this page (due to mouse input only)

UI:
- Cursor Visible
- Black Background Colour
- White Text Colour
- UI in correct positions with correct size & colours

Usability:
- Ease of Use & ease of Understanding
- Readability

Functionality:
- Do buttons turn red on hover
- Do buttons redirect using their functions correctly
- Do buttons play a sound when hovered & clicked if the settings are set

Settings Page

Normal Input: settings file
Exceptional Input: empty settings file
Extreme Input: no settings file

UI:
- Cursor Visible
- Black Background Colour
- White Text Colour
- UI in correct positions with correct size & colours