Platsorter

A platformer game incorporating sorting algorithms

Author, Ian Cen
Teacher, Mr Marsh

Project Development Report

Project Title: Platsorter

Contents

PROJECT OVERVIEW

There currently exists very few educational games focused on teaching software students about sorting algorithms. “Platsorter” aims to create a platformer style game such that the levels consist of platforms being shuffled using various sorting algorithms. The userbase will primarily consist of students who need to learn specifically about the algorithms of bubble sort, insertion sort, selection sort and bogo sort. It will be distributed by sharing the python source code files and most likely run on laptops.

DEFINING AND UNDERSTANDING THE PROBLEM

Problem Definition:

The end user will require the software to move a sprite on screen in response to the user’s inputs. Platforms displayed on screen will need to be constantly re-ordered using the bogo sort, bubble sort, insertion sort and selection sort algorithms, and the type of sort being used needs to be changed upon completion of a level. The software will also need to have a main menu where the user can start and exit the game for user friendliness. To meet these operational needs the software will require a movement and physics system, a user interface, graphical design for the player character and user interface and the 4 sorting algorithms mentioned earlier. The software will only be able to run on devices which support a python source code file. The user interface will not be any more than a start and exit button on the menu and a pause button in game. End users using the software should not be able to control the character when the game is paused or in the main menu and the character should not be able to easily clip into walls, platforms or the ground.

Feasibility Considerations – Assumptions, Dependencies and Risks:

As the game will be distributed using the python source files, this makes the game very difficult to widely distribute. However, as this project is purely educational it is unlikely and unwanted for it to be spread wider than a few students and teachers. As such the difficulty in distributing the software is not a big issue here as end users are likely to already have python installed and will only need installation of supporting libraries the software uses.

This game will contain a complex movement system in addition to animated sorting algorithms. The complexity of this task requires time and resources to be managed well to complete the task in time. This will be managed by the Gantt chart constructed in the Resource Allocation Plan.

Alternative sorting algorithm animations tend to be far superior in terms of demonstration, features, functionality and speed. This project is unique in that rather than purely simulating sorting algorithms the end user must navigate through the list, represented as column shaped platforms, to reach an objective.

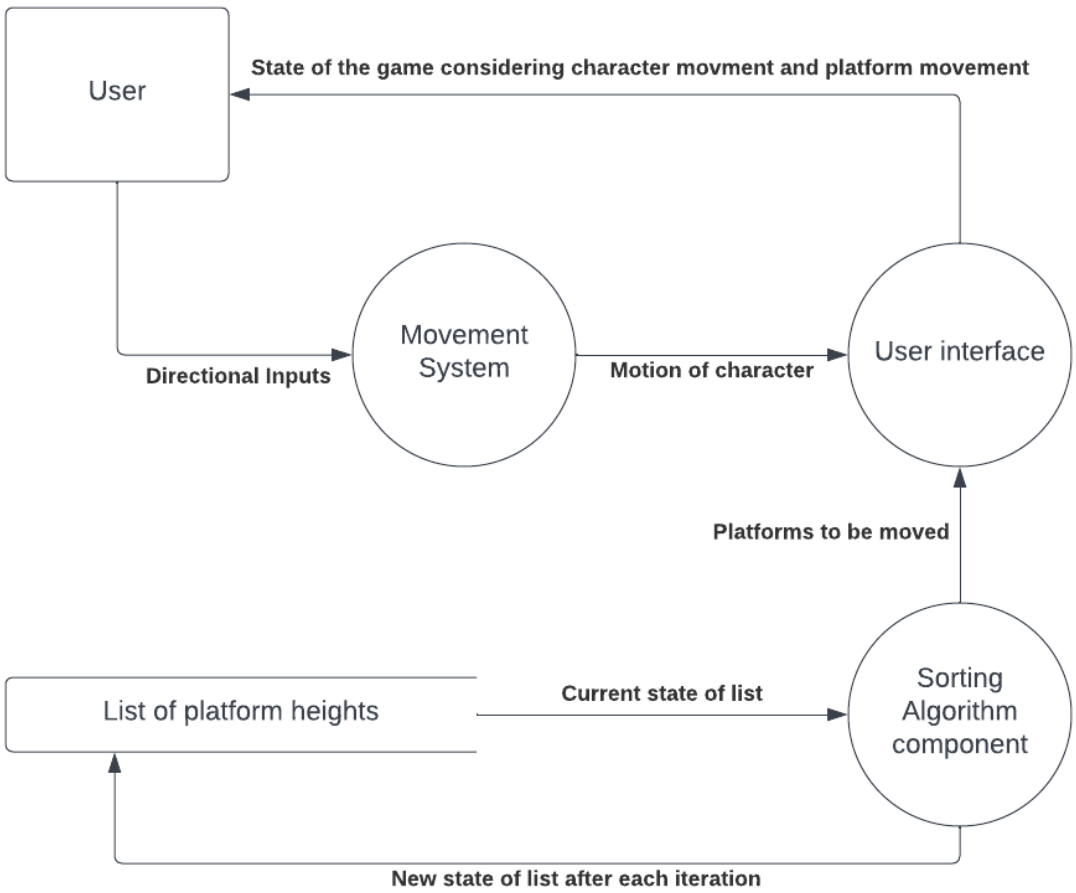
Alternative platformer games tend to also be far superior in terms of graphics, user friendliness and content, however very few platformers are educational, and no platformer game has demonstrated sorting algorithms as a main feature of the software.

The software will not store any data on end users using the software and the content within the game is not considered explicit to any ages. Checks will take place to ensure the sorting algorithms used in the software are correct representations of the advertised algorithms to not cause misinformation to end users. The software will not be copyrighted and will be allowed to copy and re-use freely. This is in case any end users wish to take some code out of this software they will not face any legal issues, and software contained within this package will not be outsourced.

PLANNING AND DESIGNING

Design Specifications/System Modelling:

Data Flow Diagram:



Movement System: This module will fully determine player motion given directional input as well as detect collisions to ensure the player does not move the character into an illegal location. After calculating where the player should be able to move this data will be sent to the user interface

User Interface: This module needs to display the data provided by the sorting algorithm component and movement system component visually so the end user is able to interpret what is happening. It will both display the location and movement of the character as well as movement of the platforms.

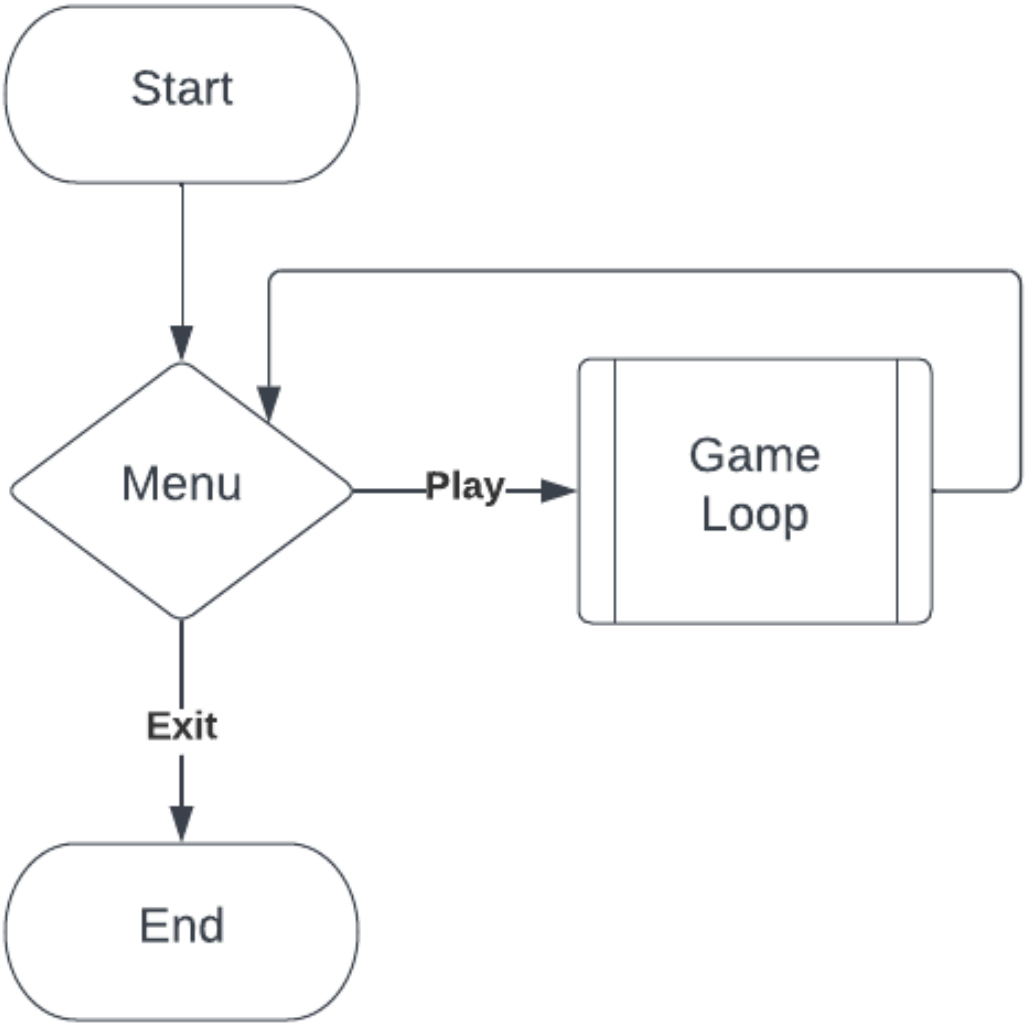
Sorting Algorithm component: This module needs to determine the sorting algorithm to use based on the current level, sort the list of platform heights in ascending order using the sorting algorithm currently selected, and communicate each swap to the user interface to display all individual swaps. Once an iteration has run through this module will need to update the list of platforms heights before beginning another iteration. If a level is to be completed this module will shuffle the list of platform heights.

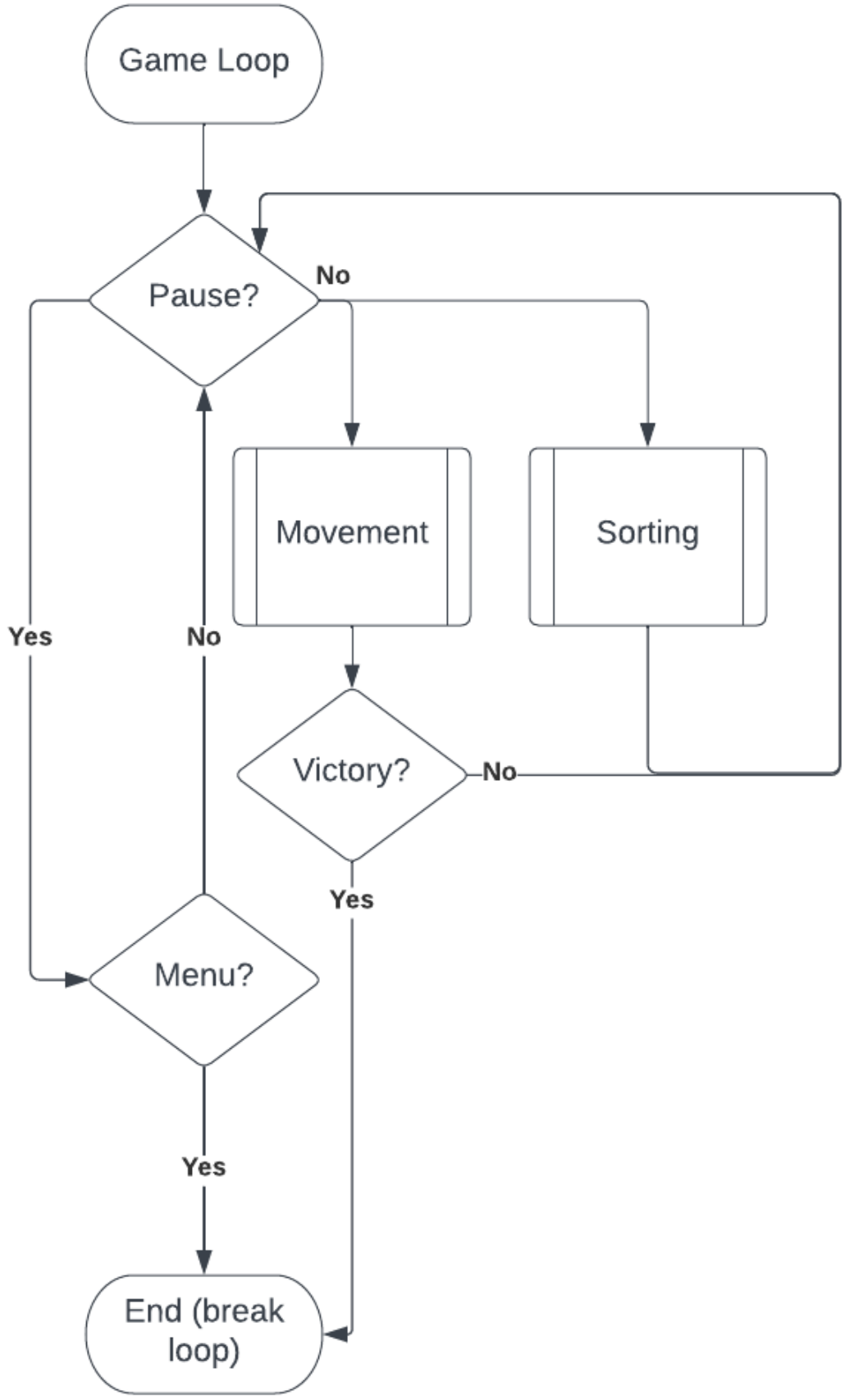
List of platform heights: This will simply be a list containing assorted values for the heights of the platforms to be generated by the interface.

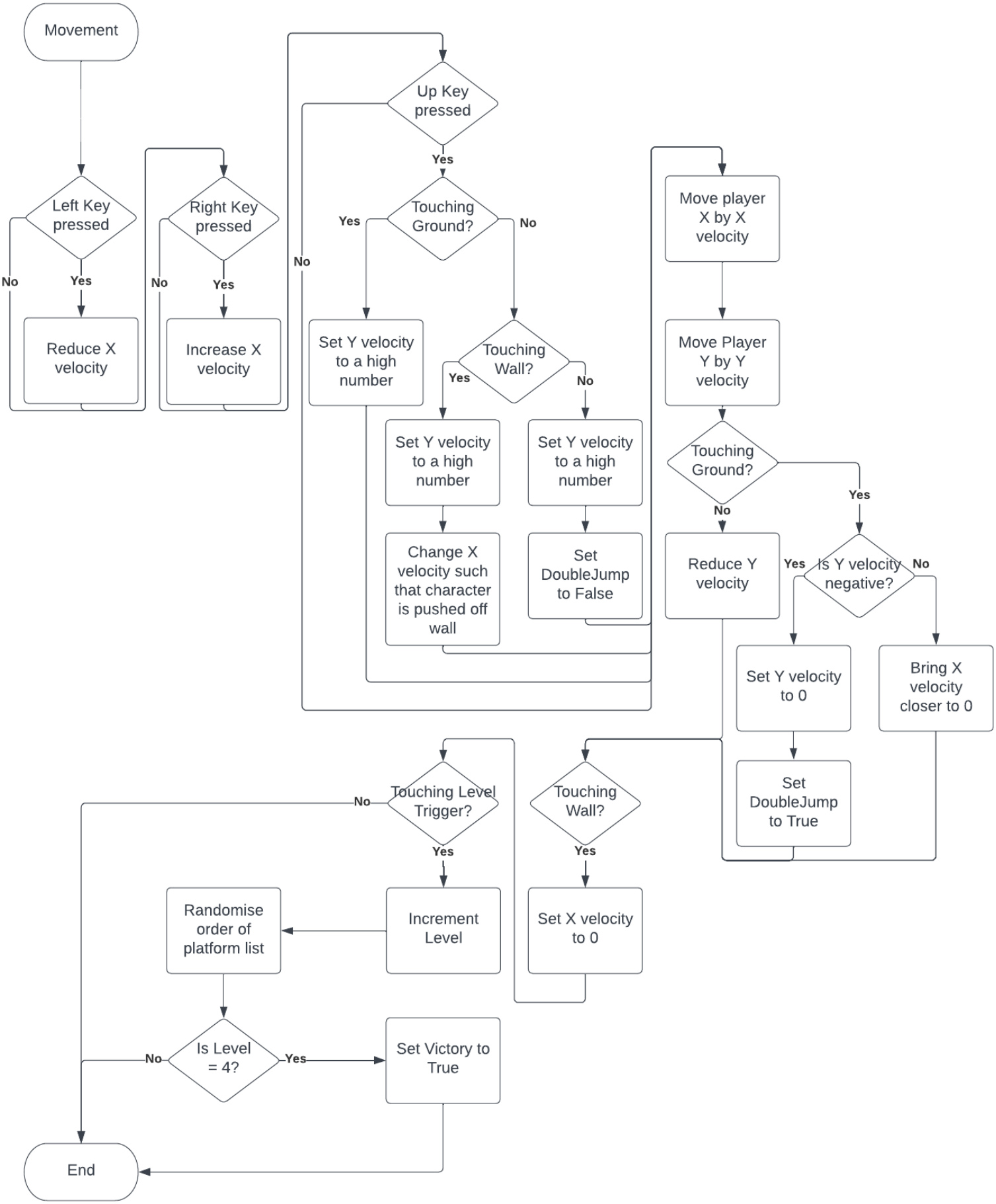
Data Dictionary:

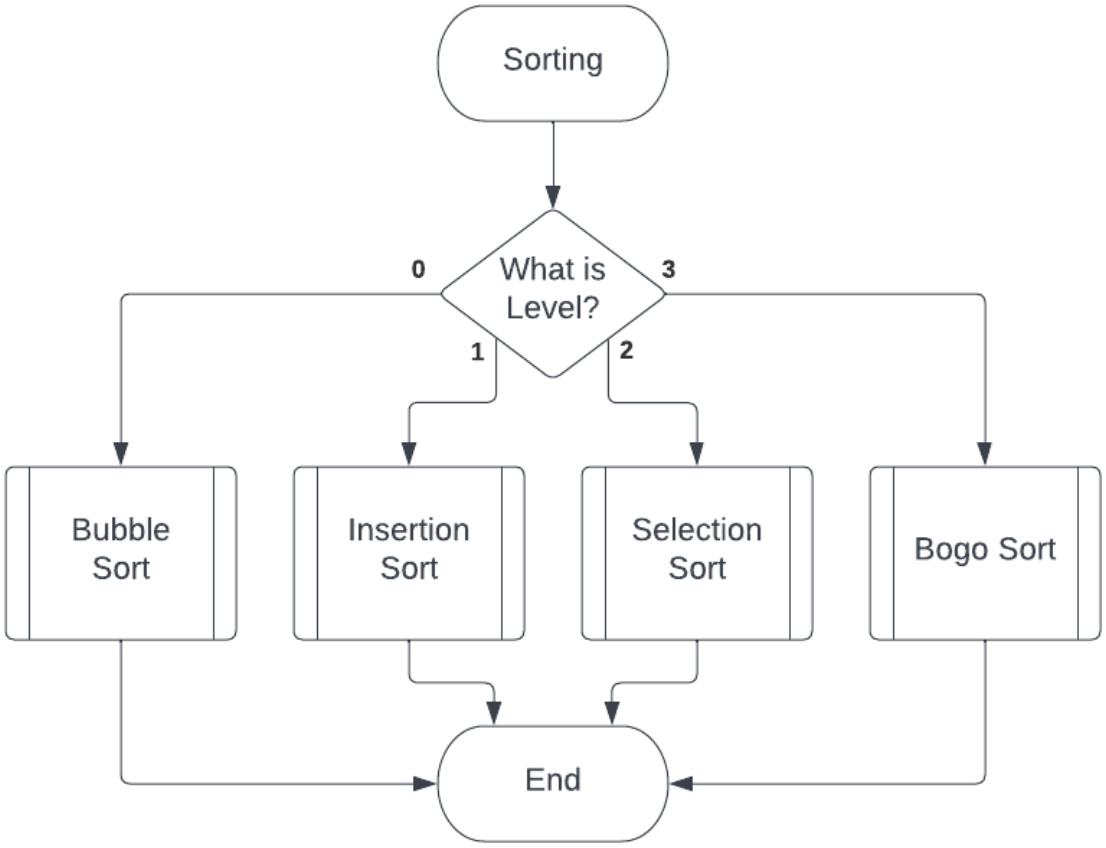
Variable Name	Data Type	Example	Description
Heights List	List	[1,5,6,2,3,4,7,8,9,10]	A list of integers. Will be shuffled around but the values should not change.
Level	integer	2	This variable will be incremented by 1 upon completion of a level and will tell the module responsible for sorting the list which sorting algorithm to use.
Touching ground	Boolean	True	This variable will tell the movement system whether the player character is touching the ground.
Pause Button Pressed	Boolean	False	Tells the user interface when to pause the game
Victory	Boolean	True	Determines if the user has beat the game
Touching Wall	string	Right	Determines if the player character is colliding with a wall, and if so, which direction.
Touching Level Trigger	Boolean	True	Determines if the player character should be advanced to the next level
Double Jump	Boolean	True	Determines if the player has used their mid-air jump.

Flowcharts:

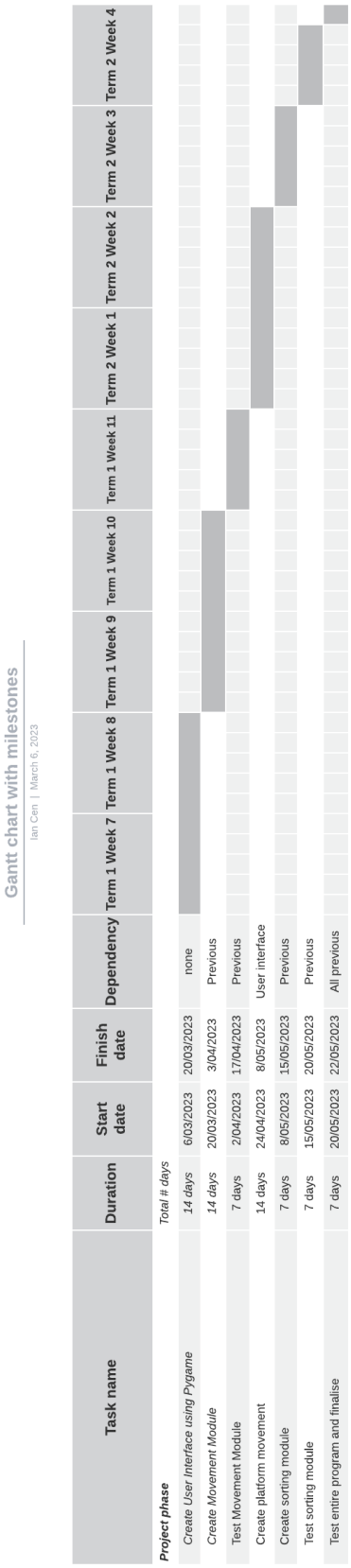



Game Loop
Pause?
No
Movement
Sorting
Yes
No
Victory?
No
Yes
Menu?
Yes
End (break loop)


Movement
Left Key pressed
No
Yes
Reduce X velocity
Right Key pressed
No
Yes
Increase X velocity
Up Key pressed
No
Yes
Touching Ground?
Yes
No
Set Y velocity to a high number
Touching Wall?
Yes
No
Set Y velocity to a high number
Change X velocity such that character is pushed off wall
Set Y velocity to a high number
Set DoubleJump to False
Move player X by X velocity
Move Player Y by Y velocity
Touching Ground?
No
Yes
Reduce Y velocity
Is Y velocity negative?
Yes
No
Set Y velocity to 0
Set DoubleJump to True
Bring X velocity closer to 0
Touching Wall?
Yes
Set X velocity to 0
Touching Level Trigger?
No
Yes
Increment Level
Randomise order of platform list
Is Level = 4?
No
Yes
Set Victory to True
End


Sorting
What is Level?
0
1
2
3
Bubble Sort
Insertion Sort
Selection Sort
Bogo Sort
End

Resource allocation Plan:



Link to clearer view of Gantt Chart: https://drive.google.com/file/d/1H_MA91KLT9Ldomw5-oV6jYHTHVTsp865/view?usp=sharing

Communication Plan:

A meeting in class will be conducted with Mr Marsh at the beginning/end of each milestone to look over what has and hasn't been done and collect feedback on what has been completed. Additionally, peer feedback will occasionally be collected in class and over text message.

Quality Assurance:

Criteria/factor	Menu Navigation	Movement System	Sorting Animation	Graphical design
Usability	Moderate	High	Low	Low
Meeting intended purpose	High	High	Moderate	High
Performance	High	High	High	High
Features	Low	Moderate	Low	Low
Functionality	High	Low	High	High

IMPLEMENTING

Implementation Strategies

All modules will be developed by Ian in a single python file. Modules will be separated by classes and functions.

Module Design

Player movement, Text, buttons, platforms, collisions, sorting algorithms and the main game loop are built separately.

Implementation Errors

Player moves slower on a platform moving upwards and moves faster on a platform moving downwards. Also, unlikely chance for player to clip through ground, however only recorded twice.

TESTING AND EVALUATION

Testing

While developing the software, modules were independently tested as they were developed.

Basic Player movement Module:

Development of this module was strongly supported by an online tutorial, as such, minimal bugs were found at this stage of development. However, the online tutorial's collision detection only applied to the top of platforms. Entering a platform from below would teleport the player above the platform.

Input	Expected Output	Output	Fix
Right arrow key	Player moves right	Expected output	None needed
Left arrow key	Player moves left	Expected output	None needed
C	Player jumps up, then falls back down	Expected output	None needed
Repeatedly pressing C	Player jumps up, then falls back down and only jumps again once hitting the ground	Player keeps jumping up again and again	Only allow player to jump if on ground
None, let player land on ground	Player does not fall through the ground	Expected output	None needed
R	Player position is reset to around the center of the screen.		None needed

Side Scrolling Module:

This module was developed because I felt the screen was not wide enough to support the number of columns I wanted to sort while also having the columns be wide enough to jump on comfortably.

Initially the scrolling module would move everything in one go, and it would be based on the player's velocity. In the final version this is changed to move everything until the player is no longer past the scroll trigger.

Input	Expected Output	Output	Fix
Moving into any screen scroll trigger	Screen scrolls left/right according to player movement	All platforms including floor and ceiling scroll	Floor and ceiling need to be excluded from scrolling
Moving towards leftmost/rightmost walls	Screen stops scrolling when player nears border walls.	Screen keeps scrolling when player nears border walls, resulting in the borders moving in.	Prevent scrolling if a border wall is projected to move too far into the screen.
Moving the player left by a very small	Screen scrolls a very small amount, then stops.	Player stays still, screen scrolls but continues scrolling.	When the player moves left a tiny amount, the integer is

amount, then releasing.		When left border comes into view test platform moves into the wall.	floored to -1, resulting in the player not moving (the player rounds to 0) but everything else moving. The solution was to truncate the x velocity when scrolling so a negligibly small velocity would not scroll the screen.
Final version of collision module implemented			
Moving into any screen scroll trigger (After reworking movement and collisions)	Screen scrolls left/right according to player movement	Screen scrolls left/right according to player movement but the player jumps back and forth.	Instead of scrolling the screen based on the player velocity, scroll the screen until the player is no longer past a scroll trigger

Collision Module:

This was arguably the most tested and reworked module out of all the modules. The initial module would only function to prevent gravity if the player collided with a platform. This was changed to detect the direction of player motion during a collision to determine which side the player collision occurred from, however this meant if the player was moving in both the x and y axis at the same time it becomes impossible to tell whether the collision is in the x axis or y axis. The collision was reworked again to calculate the gradient of the player's movement and trace the player back out of the detected collision.

Input	Expected Output	Output	Fix
Moving into any wall from the side	Player does not move into wall	Expected output	Rework collision
Jumping into a platform from below	Player does not move through platform	Player is bumped up on top of platform	Rework collision
First collision rework (Detect player movement direction)			
Jumping	Player jumps as usual	Player clips under platform because module detects player moving upwards and colliding with platform	Only move player under platform if the player is completely below the top of the platform. Moving left, right, and down are modified likewise.
Falling fast (By falling from a higher place, gravity makes the player fall faster)	Player does not fall through ground	Player falls through ground.	The player falls fast enough that the frame it collides with the ground, it has moved partially below the platform, resulting in the player being clipped under the platform.
Second collision rework (Projection tracing)			

Jumping	Player jumps as usual	Player jumps as usual	None needed
Moving into any wall	Player is stopped from moving into wall	Player is stopped from moving into the wall but tends to slide up the wall.	See below
Moving left by a very small amount	Player moves left a very small amount	Player teleports into the air, falls to the ground, then teleports into the air again, higher this time. This repeats until the player disappears from the screen.	Gradient calculation set the lesser value of x/y to be 1, and the greater value to be 1/-1 based on direction. The math would divide the greater value by the lesser value to get a larger number which would move the player in the gradient the player was predicted to move in. The problem was caused by very small values resulting in the larger value being multiplied to be larger every time. Fix was implemented by setting the larger x/y velocity value to 1/-1 and reducing the smaller x/y value accordingly.
Moving left/right while on the ground	Friction slows the player down partially, but player moves left/right	Player appears to move extremely slow, and friction seems instant. When jumping, player is launched at fast velocity as if player had been building up velocity.	The original fix implemented in the beta after detecting a ground collision was to undo all collision motion, apply friction and then redo the motion. This appeared to work due to an additional logic error which instead of undoing all the collision motion set the player to the projected position. The correct fix implemented in the final version of Platsorter reapplies movement in the

			direction the collision did not occur in.
Moving left/right while riding a platform moving up	Player moves in the direction specified	Player moves in the opposite direction	Caused by a collision still being detected even after player is moved to the starting position of that frame, resulting in the player being traced further back than the player should be, resulting in backwards movement. Fixed by applying the fix described above.
Moving into a wall while on the ground (after implementing previous fix)	Player does not move into wall or ground	Player alternates between bouncing out of the wall/ground. Not a huge hindrance to gameplay however appears weird to users (from beta feedback).	After reapplying movement in the direction the collision did not occur in, check if there is an additional collision in this direction too. If there is, move the player back like normal.

Wall collisions:
Technically a submodule of the collisions module which blends with the movement module (jumping). This code was developed alongside the collision module however I thought it would be more appropriate to list tests relating to walls separate to collisions in general.

Input	Expected Output	Output	Fix
Jumping into a wall and holding the direction key into the wall	Player slides down wall	Player sticks to wall	This was caused by movement not being reapplied in the direction the collision did not occur in. This was worked around in the beta by forcefully moving the player down some pixels when holding into a wall, however resulted in the player vibrating into the ground when moving into a wall and on the ground. Fix was applied by reapplying movement not in direction of collision.

Jumping when moving into a wall	Player performs a wall jump	Player only occasionally performs a wall jump, other times there is no response.	The player movement is based off acceleration, velocity, and displacement. When moving into a wall the player velocity in the x direction is reset to 0, meaning there are a few frames after a wall collision where the player is moving into a wall but not colliding with it. The player appears to not move into the wall on the screen however a wall collision is only detected for the frames the player has built enough velocity to trigger a collision again. The fix was implemented by checking if the player is 1 pixel away from any wall on the sides and enabling wall jump if so.

Text modules:
Text modules tended not to have that many logic errors when testing, syntax errors were more frequent and detected immediately.

Input	Expected Output	Output	Fix
Completing level 4 and moving to win screen	Text appears saying “You Win!”	Text appears for a single frame before disappearing	Text was only drawn on screen once in the game loop, resulting in the next frame covering it up immediately. To fix this, a separate section in the game loop needed to be created which would draw the text over the background surface each frame.

Sorting modules:

Input	Expected Output	Output	Fix
Completion of a level	Level is reset, platforms are reset to starting heights and begin sorting	Some platforms refuse to shrink below a certain size and appear the same height as others	When resetting platforms, the sprites created for the platforms used in the previous level were not killed, resulting in those platforms remaining. The solution was to kill the sprites of all the previous platforms before regenerating the level.

Menu modules:

Input	Expected Output	Output	Fix
Mouse hovers over button	Button changes to a lighter colour	Button changes to a lighter colour	None needed
Mouse clicks Play button	Game begins	Game Begin	None needed
Mouse clicks How to Play button	Opens a link to the user manual	Opens a link to the user manual	None needed
Mouse clicks Back to menu button	Returns to first menu screen	Returns to first menu screen, however upon pressing “Play” the level text displays bogo sort	Set the level text to selection sort when clicking play

Final Software:

Input	Expected Output	Output	Fix
Player moves while on platform moving up	Player moves like normal	Player moves but with slightly more friction than normal	Decided to leave in game due to time constraints and the low severity of the bug. (Bug is hardly noticeable)
Player moves while on platform moving down	Player moves like normal	Player moves but with slightly less friction than normal	Same as above
Player plays through the game multiple times	Game does not unexpectedly crash and handles multiple runs of the game fine	Expected output	None needed

Player enters level 1, level 2, level 3, level 4	Platforms sort according to respective sorting algorithms	Expected output	None needed
Player moves left while falling?	Player moves like normal	Player clips through platforms and disappears.	Issue has only occurred twice and am unable to reproduce. Kept reset key in final program in case.

Evaluation

The initial design of Platsorter incorporated a Pause button and ordered the levels in the order of Bubble sort, Insertion sort, Selection sort then Bogo sort. The final software does not include a pause button partially due to time constraints and also because such functionality is not needed to fulfill original requirements. The level order was switched around because from a game design perspective, Selection sort was the easiest sort to start with as it directly built a staircase. Insertion sort similarly built a staircase but worked slower. Bubble sort instead bubbles the large platforms to the right and is even slower at building a staircase. Bogo sort is unlikely to build a staircase within a feasible amount of time. Side scrolling was added in for a better user experience, allowing platforms to be thick and numerous. The software was also initially going to be distributed via source code, however by discovering PyInstaller I was able to create an executable file for easier distribution.

Movement, sorting and level progression aligns with original requirements.

Beta test feedback shows mac users would not be able to run the executable however in those cases the original source code alongside libraries can be distributed. A replay button was added to the win screen in accordance with feedback and base player speed was also increased based on feedback. Spazzing out when moving into a wall has been fixed. Movement was positively received.

After the final tests of the software, 3 main bugs remain unfixed. 2 of them (up/down platform movement) can be passed of as a feature rather than a bug. The third (unexplained player clipping) was not fixed due to time constraints and rarity in reproducing. In the future, the software could be improved by fixing these glitches, adding a pause menu, adding a timer to tell the player how fast they beat the game, different sorting animations and potentially more sorting algorithms.

MAINTENANCE OF THE SOFTWARE

Maintenance will not be conducted however the source code may be used for future projects.