Mini Project Planning Form **Brian Lee**

## Who is your application for?

The application is targeted, designed and programmed for young children e.g. siblings in a family.

Children are very active and hence lack of activities may lead to constant boredom for children. This is a problem as it is essential for children to develop cognitive skills and motions through activities, but if they are bored they are lost chances to develop. Hence the objective is to create a very simple yet entertaining game that will not only provide source of entertainment but serve as source of education to enhance the children’s cognitive skills.

## What is their current problem?

## How will your product solve this problem best?

The product provides quick, easy, cognitive entertainment that will not only get rid of their boredom but also serve as means of inducing bonding between the children/siblings. Hence the purpose of this program is to provide children a simple game of tron, an old arcade game, which will not only allow children to bond as they play against each other but also encourage them to develop their cognitive skills by learning to form different strategies to beat each other within the game.

## Success criteria

1. The program must accept users’ input simultaneously e.g. arrow keys and WASD
2. Program must hold exit button that allows quit game
3. Message indicating who won the game is displayed at the end of each game
4. Score of both players should be displayed
5. Game should correctly indicate who won the game when a player collides with tail of another player or collides with oneself.
6. The application will run on any platform with java virtual machine
7. Game should be fast paced for exciting gameplay
8. Game should hold intuitive, or otherwise very simple way to play
9. Game should hold an instruction screen.

Mini Project Design Form

# Flowchart:

In order to have an understanding of how the program will operate within the internal algorithm with user input, a flow chart has been created to graphically present the fundamental logic of this program regarding user control:

Macintosh HD:Users:BrianLee:Downloads:compsci mini IA 1.png

# UML:

Macintosh HD:Users:BrianLee:Downloads:COMPSCI.png

**Screen Designs**

Screen name: Intro

Macintosh HD:Users:BrianLee:Downloads:Screen Intro.png

Screen name: Snake (Player vs player)

Macintosh HD:Users:BrianLee:Downloads:PvP.png

Mini Project Test Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Success Criteria | Test | Action | Expected Result | Actual Result |
| The program must accept users’ input simultaneously e.g. arrow keys and WASD | Check the WASD/arrow keys are accepted as appropriate input | Press the WASD keys and the arrow keys | W – Changes the direction of Blue player to UP  A - Changes the direction of Blue player to LEFT  S - Changes the direction of Blue player to DOWN  D - Changes the direction of Blue player to RIGHT   - Changes the direction of Red player to UP   - Changes the direction of Red player to LEFT   - Changes the direction of Red player to RIGHT   - Changes the direction of Red player to DOWN | Positive. Each input was accepted by program to make respective changes. |
| Program must hold exit button that allows quit game | Check that the program holds a functioning exit button | Press the exit button | System will close when the button is clicked. | Positive. The program holds “Exit” button that causes the program to close when clicked. |
| Message indicating who won the game is displayed at the end of each game | Check that the system displays message at the end of each game | Allow one player to win a game. Check for both cases of red and blue. | If a player wins, a message correctly indicating either red/blue won will be displayed. | Positive. “Blue player wins!” popped up in the case of red player losing, and “Red player wins!” popped up in the case of blue player losing. |
| Score of both players should be displayed | Check that the score of two players are always displayed on the screen. | Continue playing the game, observe if the scores are kept correctly and always displayed | A scoreboard showing the red player’s score, and also displaying the blue player’s score. | Positive. Scores of both players are accurately displayed on top of the screen. |
| Game should correctly indicate who won the game when a player collides with tail of another player or collides with oneself. | Check that the game displays appropriate message after a player wins the game. | Allow the red player to win. | The message should correctly show if red or blue player won the game. | Positive. The game indicates blue’s victory when the blue player wins, and indicates red’s victory when the red player wins. |
| Allow the blue player to win. |
| The application will run on any platform with java virtual machine | Check that the game can run on any java virtual machine | Play the application on different computers that all hold the Java virtual machine. | The program should flawlessly run on every platform. | Positive. The program ran fine on a computer with java virtual machine. However, there may be more ways to test this as it has been tested on only one platform. |
| Game should be fast paced for exciting gameplay | The game should feel exciting for children to play. | Have multiple clients in target group to play the game, and rate the excitement and fun they got while playing. | In order to achieve the success criteria, the application should receive decent or high ratings from the testers. | Positive. Multiple classmates, and they all showed positive comments towards the game they have played. |
| The game should have simple quick gameplay. |
| Game should hold intuitive, or otherwise very simple way to play | The game’s controls should be basic that it won’t require any or only short instruction. | Instruction should be less than 3 lines long. | Instruction should be short, and the tests should indicate that the game is relatively easy to play. | Positive. Total instruction is 2 lines, and the classmates who have played the game have almost intuitively played the game, some of them managing to play it even before the instructions were included. |
| Have clients in target group to play the game and rate the difficulty of controls. |
| Game should hold an instruction screen. | In order for players to know how to play the game, there should be basic instruction screen when application is opened. | Open the game; check whether the instruction screen opens. | Instruction screen appears when game is running. | Positive. An intro screen with instructions, play button, and exit button is displayed. |

Mini Project Class Demo Checklist

You will have 3 minutes to present your project to the class. You need to be fast, but you must include the following:

|  |  |  |
| --- | --- | --- |
|  | ✔ | Comments |
| Program opens | ✔ |  |
| Allows data input and produces output (using real screens if gui) | ✔ |  |
| OOP techniques used shown (showing code) | ✔ |  |
| Show any existing tools used (which libraries etc ) - code and/or screen | ✔ |  |
| Any GUI/file handling/other advanced technique shown | ✔ |  |
| General Comments:  Fix the bug that occurs when players come to a draw (It has already been fixed) | | |

Mini Project Evaluation

# Evaluation of success criteria

|  |  |  |  |
| --- | --- | --- | --- |
| Success Criteria | Evidence | User/buddy feedback | Further comment |
| The program must accept users’ input simultaneously e.g. arrow keys and WASD | Testing shows that all inputs are accepted and implemented appropriately by the internal algorithms of the program. | - | There isn’t a lot that can be improved on the input them, but the implication of the inputs may be enhanced. For instance The internal algorithm may be modified to allow more precise change in direction, rather than simple up/down/left/right directions, such as diagonal directions. |
| Program must hold exit button that allows quit game | Testing shows that “Exit” performs as desired by exiting the program when pressed. | Possible a new exit button on the game screen as well, purely for convenience. | As the buddy feedback shows, the program may be modified to hold an exit for each screen for more convenience. |
| Message indicating who won the game is displayed at the end of each game | Testing shows that message for both players’ victory are displayed correctly at end of each game. | - | - |
| Score of both players should be displayed | Testing proves that score of both players are displayed at all times, accurately indicating each player’s score. | - | A more easily recognizable scoreboard could be used, for instance different color and larger font. |
| Game should correctly indicate who won the game when a player collides with tail of another player or collides with oneself. | Testing shows message indicating the winner appears at the end of each game | If given more time the program should be improved so that there are more graphical features at the end of each game e.g. sound | Not only sound but also animation could be implemented if the code was to be modified. |
| The application will run on any platform with java virtual machine | Testing shows program works on platforms with java virtual machine. | - | It has been tested on mac, but it hasn’t been tested whether it runs on other computer systems with java virtual machine installed. Hence further testing should be made. |
| Game should be fast paced for exciting gameplay | Students who have played the game showed positive comments about how fun and/or exciting the game is to play. | However, they also commented by saying the game should be developed to allow more players are imply the “boost” system to allow players to raise the speed of their characters temporarily, and hence increase the pace of the game. | - |
| Game should hold intuitive, or otherwise very simple way to play | Testing indicates the game’s controls are very basic and easy. | - | - |
| Game should hold an instruction screen. | Testing appears to show that game successfully displays an instruction screen at the beginning. | Instruction screen could have more user-friendly aesthetics. E.g. larger text. | More user-friendly features such as diagrams could be used to display instructions in graphical form. E.g. the pacman is a good example: |

# What went wrong/could go better/ Improvements?

The program does not hold any major flaws that affect the accessibility of the game, but there are some flaws and possible areas of improvement within the program. Lack of encapsulation is one of the flaws. The program does hold multiple of variables as private, but majority of the variables are kept public, and hence it lacks encapsulation. Changing the access modifiers to private and declaring getter methods for each variable could easily solve this.

Apart from encapsulation multiple improvements may be made if this project was to be carried on, ultimately to IA. The code may implement a lot of changes regarding graphics and control. At current stage the aesthetics of the game is very basic, using simple rectangular shapes and plain colors, while also lacking visual effects. This may be improved by implying smoother shapes and different colors. Furthermore extended development could be used to allow implementations of visual effects. Secondly, the game currently uses WASD and arrow keys, which allow input of basic directions (up, down, left and right). Improvement in the range of possible user input could enhance gameplay and therefore bring greater entertainment. For instance more keys that allow diagonal directions could be added as an arbitrary improvement. Lastly what was mentioned in the buddy feedback was lack of sound, which affected the gameplay as the lack of sound effects made the game less exciting. Hence implication of sound may be made as improvement. Specifically, this may mean addition of a new class ‘Sound’ for utilization of sound effects.

# Bibliography

1. "2600 Connection." *2600 Connection*. N.p., n.d. Web. 15 May 2016.