Programming Project

Project Design Sheet

Grid Based Game

# Review of at least 3 different programs

## MINESWEEPER

Minesweeper is a simple grid based game that has easy to learn mechanics and is a test of logical and general reasoning. The idea of the game is that based on 3 different grid sizes representing a difficulty increase there is an assortment of randomly placed ‘mines’ on randomly selected grid tiles and the player must locate and ‘flag’ them in order to win the game. If a tile is selected that has no mine, it will either clear a large area of no mines or will display a number between 1 and 8 which represents the amount of mines adjacent to the revealed square. If a mine is clicked on then the game is over and the player has lost.

While this game would be very possible to create using C++ it is lacking in depth and is not necessarily challenging enough for a detailed write up. Additional depth could be added by asking the player to select different grid size and the amount of mines however this could lead to less enjoyment from the game if the player accidently makes it too easy or too hard.

## TETRIS

Tetris is also a relatively simple grid based game that is also mentally stimulating in a different way to Minesweeper. In Tetris the player’s ability to quickly problem solve and find the most efficient way under pressure. The key mechanic of Tetris is that different coloured shapes fall from the top of the grid down to the bottom and they stack. The objective of the game is to not let the shapes stack all the way to the top which is achieved by stacking different shapes of the same colour, after a certain amount of the same color are stacked the shapes will disappear and the player will acquire points. High Scores are displayed after every game finishes and when beaten the new High Score will be displayed.

This game is a better idea than Minesweeper in my opinion for this project as it involves many different ideas, a file to store High Scores which would be written and read from, a generation algorithm for the different shape and random colour selection, etcetera. The project is far more challenging than Minesweeper and could provide an interesting challenge to create in C++.

## PAC MAN

Pac Man is a maze based game that takes some simple concepts and adds difficulty in the form of enemies and items that must be collected in order to progress. The main goal of Pac Man is to eat all the pellets in the grid in order to progress to the next level while avoiding enemies and acquiring points. Pac Man is more complicated than the previous grid based games in my opinion as pick ups can allow for the enemies to suddenly become available to eat in order to gain more points. Similar to Tetris High Scores are displayed telling the player how many points they gathered in comparison to other players. The enemies in Pac Man take the form of Ghosts which chase the player throughout the maze making it more difficult to complete the goal and adding a challenging dynamic to otherwise standard gameplay.

The idea of creating a Pac Man style game within C++ is somewhat daunting and would provide a high level of challenge to my programming ability. A simpler maze style game however would be more in line with what I would feasibly be able to achieve where a maze can be randomly generated by an algorithm and the player would have to solve the maze within a given time or try to avoid an enemy seeker and reach the goal.

# Program Requirements and Specification

In order for my program to offer a suitable level of difficulty and be an interesting experience certain requirements both technical and gameplay wise must be established.

## GAMEPLAY REQUIREMENTS

* Player start must be sufficiently far away from goal
* Stopwatch will displayed to player at the end of each level
* High Score will be based on how many ‘Levels’ are completed
* Each subsequent level will be harder than the previous in some way
* Must be a way to track where the player has gone to minimize backtracking
* Maze must be possible to complete
* Possibly have music playing and audible cues when moving or reaching goal
* Chaser character to put pressure on player to solve maze rapidly
* There will be loops within the maze where the player can ‘outwit’ the chaser

## TECHNICAL SPECIFICATION

* The program will have a document storing High Score values which can be written to and read from, this will mean that the program will not have to be recompiled every time a High Score is beaten.
* An algorithm will be used to generate the maze layout, it will increase in size after each subsequent level
* The elements of the maze will be created based on classes so that it is easier to create additional maze levels.
* Stopwatch
* Player start and end cannot be too close together, can be defined within algorithm
* Possible algorithm to solve the maze if the player cannot come up with a solution
* Player can be controlled using keyboard keys
* Program a chaser that follows the player after a certain amount of time has passed on each level
* WSAD will be used to control the player controlled character
* Adjustment to algorithm to create ‘backtracking’ areas so the chaser can be outwitted

# Comparison to other Programs

With my program I intend to cherry pick ideas and mechanics that I think are particularly relevant or clever and would mesh well with my game idea, for example the idea of an enemy type from Pac Man and the random aspect of Minesweeper will, in my opinion create a more entertaining play experience than simply a game where you must reach the end of a pre made maze. The addition of high scores will add a level of competitively that will encourage players to keep replaying the game to try and beat their or other peoples previous best scores. The inclusion of an enemy type will add pressure to the player and create a sense of tension that any wrong turn could potentially be game over.

# Program Decomposition

Main menu – A main menu will have options to “Start the Game”, “View High Scores” and “Exit”

Generating Maze Layout – A maze generating algorithm will be used which will create challenging scenarios and will also include ‘backtracking’ areas where the AI chaser can be outwitted

Chaser AI – Chaser will not appear in the game until level 3 so the player is not bombarded with mechanics. Another algorithm will be used that will solve every maze, the chaser will follow this algorithm and if it notices the player has diverted it will chase him and try to corner him into a dead end

Player/Goal – Both elements will have to be a reasonable distance apart so that any maze is not too easy, both will have to be actually accessible to each other and a situation where a maze is ‘unsolvable’ must not occur. Everywhere the player goes will be marked by colouring the tile that was moved from in an incrementing gradient of colour like so.



# Psuedocode

MAIN CODE

MAIN MENU

START GAME

initialize first maze

START GAME LOOP

initialize player in random position

initialize goal in random position

initialize chaser within certain area of player

if player reaches goal

initialize next level

if player gets caught

game over

if game over

display high score

if playerScore > highscores

highscores = playerscore

go back to menu

end game loop

DISPLAY HIGH SCORES

SHOW HIGH SCORES

EXIT

EXIT PROGRAM

PLAYER CODE

WASD CONTROLS

AS A MOVE IS MADE COLOUR IN TILE THAT WAS MOVED FROM TO INDICATE PREVIOUS TRAIL

CHASER CODE

AI CONTROLLED

MOVE AT A SET SPEED

WILL GO TOWARDS GOAL AND IF IT NOTICES PLAYER IS NOT GOING THE SAME WAY WILL BACKTRACK AND HUNT HIM

MAZE CODE

INCREASE IN SIZE AFTER EVERY INITIALISATION

WHEN CHASER IS INTRODUCED MAZE WILL CREATE LOOPING AREAS SO THE PLAYER CAN DODGE THE CHASER

MAZE GENERATING ALGORITHM