LIST OF FIGURES

Figure 1.1: Number of major commercial games released per year on various game platforms (excluding Android) in 1975-2012

Figure 1.2: Application Areas of AI

Figure 2.1: A graph Example

Figure 2.2: A directed graph example

Figure 2.3: Depth-First Search Node-Visiting Process

Figure 2.4: DFS pseudo code

Figure 2.5: Recursively defined DFS

Figure 2.6: Breadth First Search (BFS) algorithm

Figure 2.7: BFS pseudo code

Figure 2.8: Opening an Empty Tile in Minesweeper

Figure 2.9: Turn based tactic / strategy games

Figure 2.10: Incorrect Results in Limiting DFS Range

Figure 3.1: Components of .NET Framework

Figure 3.2: Game editor of the Unity3D game engine

Figure 3.3: Game Objects in Unity3D

Figure 3.4: Toobar window in Unity 3D

Figure 3.5: Profiler window in Unity 3D

Figure 3.6: Profiler Controls

Figure 3.7: Profiler Timeline

Figure 3.8: Animation in Unity 3D

Figure 3.9: Project View

Figure 3.10: Third Person camera setting

Figure 3.11: Vector between camera and objecting position

Figure 3.12: Bounding Box

Figure 3.13: Bounding Sphere

Figure 4.1: Graphical representation of state S to goal state G

Figure 4.2: Process for the H value for each node

Figure 4.3: Calculation of G value

Figure 4.4: Child node and Parent node

Figure 4.5: Calculating F value

Figure 4.6: Shortest Path using designed and developed algorithm

Figure 4.7: Novel Algorithms optimal results

Figure 4.8: The pseudo code

Figure 4.9: New designed and developed path finding algorithm's graphical representation from starting node (red) to target node (blue)

Figure 4.10: Storing last sighting values

Figure 4.11: Implementation of trigger for changing player’s state

Figure 4.12: Creation of new instance of a graph type

Figure 4.13: Implementation of function Creategraph()

Figure 4.14: Adding node into the graph

Figure 4.15: Implementation of addgraph()

Figure 4.16: Implementation of removegraph()

Figure 4.17: Steering behavior for seek
Figure 4.18: Obstacle avoidance………………………………………………… 142
Figure 4.19: Shows representation of obstacle avoidance example…… 144
Figure 5.1: Relationships between units in a game with broken game balance…………………………………………………………………… 147
Figure 5.2: 3D model in developing state made by 3D artist……………… 148
Figure 5.3: Model made using 3D max……………………………………… 149
Figure 5.4: Unwrapping the model………………………………………… 150
Figure 5.5: 3D model after applying material (texture)…………………… 151
Figure 5.6: Default FBX exporter options…………………………………… 153
Figure 5.7: Lightmapping setting…………………………………………… 155
Figure 5.8: Alternate material setup………………………………………… 156
Figure 5.9: Screenshots of menu……………………………………………… 157
Figure 5.10: Setting up raycasting…………………………………………… 158
Figure 5.11: Flock AI…………………………………………………………… 159
Figure 5.12: Flock AI in flying state………………………………………… 160
Figure 5.13: Flock AI flying in a group……………………………………… 161
Figure 5.14: Code for flock AI part 1……………………………………….. 162
Figure 5.15: Code for flock AI part 2……………………………………….. 163
Figure 5.16: Code for flock AI part 3………………………………………… 164
Figure 5.17: FSM for enemy………………………………………………….. 166
Figure 5.18: Enemy running towards player……………………………… 167
Figure 5.19: Combat between player and enemy…………………………… 168
Figure 5.20: Screen shots of player dead state…………………………….. 169
Figure 5.21: Code for combat AI system and implementation of animation parameter………………………………………………………… 170
Figure 5.22: Code for combat system using raycasthit()……………….. 171
Figure 5.23: NPCs on the terrain…………………………………………….. 173
Figure 5.24: NPCs changed direction while collision occurs……………… 174
Figure 5.25: Code for NPCs part 1………………………………………….. 174
Figure 5.26: Setting up path using current position………………………… 176
Figure 5.27: Code for NPCs part 2………………………………………….. 176
Figure 5.28: Screen shots of crowd behaviour for pedestrian…………….. 177
Figure 5.29: Screen shots of game for demonstrating pedestrian behaviour……………………………………………………………………………… 178
Figure 5.30: Screenshots of pedestrian moving towards directions……… 179
Figure 5.31: Crowd behaviour and pedestrian code………………………. 180
Figure 6.1: Execution of Dijkstra algorithm scans all nodes……………… 185
Figure 6.2: Execution of newly designed and developed algorithm which is more efficient than Dijkstra……………………………………… 186