Introduction

1.1 Introduction
1.2 3D Games and Gaming Industry
1.3 Motivation
1.4 Problem Statement
1.5 Objectives
1.6 Structure of Thesis
1.7 Summary
Chapter 1: Introduction

1.1 Introduction
This chapter provides the information about 3D games, gaming industry and latest technology used in 3D game development. 3D game technology is enormously large Technology field. It also covers overview about the technology and AI used in the 3D game development. The first part of this chapter discusses about 3D games and gaming industry. The later part of the chapter discusses about Artificial Intelligence and Artificial Intelligence in 3D games.

1.2 3D Games and Gaming Industry
A Computer game or a video game is an interactive application with the main purpose of entertaining the user (player). Initially games were developed by computer scientists primarily to entertain themselves and their colleagues. The history of video games goes as far back as the early 1950s, when academics began designing simple games, simulations, and artificial intelligence programs as part of their computer science research. In late 1970s, with immense increase in computing power, graphical capabilities and mass production of personal computers, game development became an industry.

Competition between game developers and rapidly increasing expectations of players forced developers to make game as computationally complex and as graphically appealing as was allowed by the average hardware of the end user [1, 24].

The 1990s saw a vast improvement of the presentation of games: from a palette of just 4-8 colors and pixelated characters in the beginning of the 1990s, evolving into 32bit-colored masterpieces of 2D graphics by the late 1990s. A transaction from 2D to 3D graphics quickly followed. Today
Chapter 1: Introduction

A commercial game takes 2-3 years development time and 20-50 developers.

In late 1990s with worldwide adoption of cell phones, a part of development effort of game companies got focused on making small, low budget 2D games for these new devices. These games weren’t really successful due to limited distribution possibilities, insufficient user acceptance of payment methods, extreme fragmentation of hardware and low performance of the java MIDP platform which cell phones used for games [24].

In 2007 a new device called iPhone by Apple inc. turned the peripheral niche of mobile games into an extremely profitable space for both small indie developers and large companies. Apple provided a single distribution channel (AppStore) which the user could access directly and easily from the device, a stable standardized development platforms, and a quickly growing audience of players who were ready to pay for games [1].

The iPhone runs an operating system iOS which is based on Apple’s operating system Macintosh for laptops and desktop computers. A new term, “smartphone” was coined to describe mobile phones with an approximately 5-inch large touch sensitive display and more powerful graphical and computational capabilities than a usual mobile phone[1].

In 2009 a competing operating system Android by Google Inc. was released and by 2011 outgrew iOS in popularity due to being open-source and less restrictive. Figure 1.1 represents the number of games developed for smartphones grew at an unprecedented rate during 2009-2012. At the moment of this writing mobile games are the only growing market of video games while all other markets (desktop games, gaming consoles, web-browser games, games for handheld devices) are either stagnating or shrinking[1, 24].
Chapter 1: Introduction

Artificial Intelligent Game System for Steer Behavior

The number of smartphones is projected to reach 2 billion by 2015[1, 33]. These devices have powerful graphical capabilities that allows them to run games that are technically comparable to those of mid-late 1990s for desktop computers. Application stores (store for windows, AppStore for iOS and Google Play for Android) allow users to easily download and install these games. Integrated payment systems allow a customer to seamlessly buy games or make in-game purchases. All these factors make smartphones a strategically vital platform for game development studios. Many game programmers have to switch from older platforms like Windows and older game development APIs like DirectX to new mobile platforms[24].

Figure 1.1: Number of major commercial games released per year on various game platforms (excluding Android) in 1975-2012[1]
The market of smartphone games was initially dominated by games for casual players—players who don't or very occasionally play games on a computer and don't want games to be challenging. However, games on mobile platforms started their evolution the same way they did on desktop computers but at a much faster rate. Games are getting more complex, with more content and 3D graphics. As the casual gamers become more experienced in gaming, they ask for more challenging games, closer to those hardcore gamers play on computers. Genres common on desktop games attempt to conquer mobile platforms and adapt to the limitations these platforms impose.

One such genre is that of the ACTION where the player controls and destroy the enemy, gathers resources in real time with the goal to destroy opponents. This genre was very popular in late 1990s, and slowly became a niche market in 2000s and early 2010s. AI of Steer behavior is one of the important factor specially in 3d ACTION games[1,2,4].

Development of computer game requires collaboration of developers in many areas including programming, 2D and 3D art, music and sound effects, game design and testing. Programming tasks while developing a game are divided into a few areas including AI and gameplay, networking, tools and physics.

Initially games were designed by programmers themselves: they were both think up the idea of the game and implementing it. This is still the case for indie games and games done by very small teams (2 or 3 developers). Since the 2000s the games have become so large in content that need manifested to have special team members doing game design only [1,2,3].
Chapter 1: Introduction

Game designers write the design documents of the game—a large document which in the games industry acts as a functional specification document.

Game designers also fill the game with content: populate levels with monsters, tweak the game rules and parameters of in-game elements and write in-game texts (for example, conversations of characters, tutorial, quest assignments.) with game content or game resources programmers refer to images, sounds, in-game texts, 3D models, scripts and other assets which are specific to the currently developed game and are usually produced by non-programmers. Artificial Intelligence (AI) is required for this purpose.

1.2.1 Artificial Intelligence

In 3D video games, artificial intelligence is used to produce the illusion of intelligence in the behavior of non-player characters (NPCs). The techniques used typically draw upon existing methods from the field of artificial intelligence (AI). However, the term game AI is often used to refer to a broad set of algorithms that also include techniques from control theory, robotics, computer graphics and computer science in general. Figure 1.2 represents Application Areas of AI[2].
The subject of artificial intelligence spans a wide horizon. It deals with various kinds of knowledge representation schemes, different techniques of intelligent search, various methods for resolving...
uncertainty of data and knowledge, different schemes for automated machine learning and many others[2].

The most common areas of AI includes Expert systems, Game-playing, Theorem-proving, Natural language processing, Image recognition, Robotics and many others. The subject of artificial intelligence has been enriched with a wide discipline of knowledge from Philosophy, Psychology, Cognitive Science, Computer Science, Mathematics and Engineering. Thus as Figure 1.2 shows, they have been referred to as the parent disciplines of AI and at-a-glance look at figure 1.2 also reveals the subject area of AI and its application areas[2].

The subject of AI was originated with game-playing and theorem-proving programs and was gradually enriched with theories from a number of parent disciplines. As a young discipline of science, the significance of the topics covered under the subject changes considerably with time. At present, the topics which we find significant and worthwhile to understand this subject are outlined bellow[1,2]:

**Learning Systems**

Among the subject areas covered under artificial intelligence, learning systems needs special mention. The concept of learning is illustrated here with reference to a natural problem of learning of pronunciation by a child from his mother. The hearing system of the child receives the pronunciation of the character “A” and the voice system attempts to imitate it. The difference of the mother’s and the child’s pronunciation, hereafter called the error signal, is received by the
child’s learning system auditory nerve, and an actuation signal is generated by the learning system through a motor nerve for adjustment of the pronunciation of the child. The adaptation of the child’s voice system is continued until the amplitude of the error signal is insignificantly low. Each time the voice system passes through an adaptation cycle, the resulting tongue position of the child for speaking “A” is saved by the learning process. The learning problem discussed above is an example of the well-known parametric learning, where the adaptive learning process adjusts the parameters of the child’s voice system autonomously to keep its response close enough to the “sample training pattern”. The artificial neural networks, which represent the electrical analogue of the biological nervous systems, are gaining importance for their increasing applications in supervised (parametric) learning problems. Besides this type, the other common learning methods, which we do unknowingly, are inductive and analogy-based learning. In inductive learning, the learner makes generalizations from examples. For instance, noting that “cuckoo flies”, “parrot flies” and “sparrow flies”, the learner generalizes that “birds fly”. On the other hand, in analogy-based learning, the learner, for example, learns the motion of electrons in an atom analogously from his knowledge of planetary motion in solar systems[1,2].

**Knowledge Representation and Reasoning**

In a reasoning problem, one has to reach a pre-defined goal state from one or more given initial states. So, the lesser the number of transitions for reaching the goal state, the higher the efficiency of the reasoning system. Increasing the efficiency of a reasoning system thus requires minimization of intermediate states, which indirectly calls for
an organized and complete knowledge base. A complete and organized storehouse of knowledge needs minimum search to identify the appropriate knowledge at a given problem state and thus yields the right next state on the leading edge of the problem-solving process. Organization of knowledge, therefore, is of paramount importance in knowledge engineering. A variety of knowledge representation techniques are in use in Artificial Intelligence. Production rules, semantic nets, frames, filler and slots, and predicate logic are only a few to mention. The selection of a particular type of representational scheme of knowledge depends both on the nature of applications and the choice of users [2].

**Planning:**

Another significant area of artificial intelligence is planning. The problems of reasoning and planning share many common issues, but have a basic difference that originates from their definitions. The reasoning problem is mainly concerned with the testing of the satisfiability of a goal from a given set of data and knowledge. The planning problem, on the other hand, deals with the determination of the methodology by which a successful goal can be achieved from the known initial states. Automated planning finds extensive applications in robotics and navigational problems [2].

**Knowledge Acquisition**

Acquisition (Elicitation) of knowledge is equally hard for machines as it is for human beings. It includes generation of new pieces of knowledge from given knowledge base, setting dynamic data structures for existing knowledge, learning knowledge from the
Chapter 1: Introduction

environment and refinement of knowledge. Automated acquisition of knowledge by machine learning approach is an active area of current research in Artificial Intelligence [2].

**Intelligent Search**

Search problems, which we generally encounter in Computer Science, are of a deterministic nature, i.e., the order of visiting the elements of the search space is known. For example, in depth first and breadth first search algorithms, one knows the sequence of visiting the nodes in a tree. However, search problems, which we will come across in AI, are non-deterministic and the order of visiting the elements in the search space is completely dependent on data sets[1,2].

**Logic Programming**

For more than a century, mathematicians and logicians were used to designing various tools to represent logical statements by symbolic operators. One outgrowth of such attempts is propositional logic, which deals with a set of binary statements (propositions) connected by Boolean operators. The logic of propositions, which was gradually enriched to handle more complex situations of the real world, is called predicate logic. One classical variety of predicate logic-based programs is Logic Program. PROLOG, which is an abbreviation for Programming in Logic, is a typical language that supports logic programs. Logic Programming has recently been identified as one of the prime area of research in AI. The ultimate aim of this research is to extend the PROLOG compiler to handle spatial -temporal models and support a parallel programming environment. Building architecture for PROLOG machines was a hot topic of the last decade [1, 2].
Soft Computing

Soft computing, according to Prof. Zadeh, is “an emerging approach to computing, which parallels the remarkable ability of the human mind to reason and learn in an environment of uncertainty and imprecision”. It, in general, is a collection of computing tools and techniques, shared by closely related disciplines that include fuzzy logic, artificial neural nets, genetic algorithms, belief calculus, and some aspects of machine learning like inductive logic programming. These tools are used independently as well as jointly depending on the type of the domain of applications [1,2].

Management of Imprecision and Uncertainty

Data and knowledge bases in many typical AI problems, such as reasoning and planning, are often contaminated with various forms of incompleteness. The incompleteness of data, hereafter called imprecision, generally appears in the database for i) lack of appropriate data, and ii) poor authenticity level of the sources. The incompleteness of knowledge, often referred to as uncertainty, originates in the knowledge base due to lack of certainty of the pieces of knowledge. Reasoning in the presence of imprecision of data and uncertainty of knowledge is a complex problem. Various tools and techniques have been devised for reasoning under incomplete data and knowledge. Some of these techniques employ i) stochastic ii) fuzzy and iii) belief network models. In a stochastic reasoning model, the system can have transition from one given state to a number of states, such that the sum of the probability of transition to the next states from the given state is strictly unity. In a fuzzy reasoning system, on the other hand, the sum of the membership value of transition from
the given state to the next state may be greater than or equal to one. The belief network model updates the stochastic / fuzzy belief assigned to the facts embedded in the network until a condition of equilibrium is reached, following which there would be no more change in beliefs. Recently, fuzzy tools and techniques have been applied in a specialized belief network, called a fuzzy Petri net, for handling both imprecision of data and uncertainty of knowledge by a unified approach[2].

1.2.1.1 AI Algorithm

Formally speaking, an artificial intelligence algorithm generally means a non-conventional intuitive approach for problem solving. The key to artificial intelligence approach is intelligent search and matching. In an intelligent search problem / sub-problem, given a goal (or starting) state, one has to reach that state from one or more known starting (or goal) states [35].

For example, consider the 4-puzzle problem, where the goal state is known and one has to identify the moves for reaching the goal from a pre-defined starting state. Now, the less number of states one generates for reaching the goal, the better. That is the AI algorithm.

The question that then naturally arises is: how to control the generation of states? This can be achieved by suitably designing control strategies, which would filter a few states only from a large number of legal states that could be generated from a given starting / intermediate state.

As an example, consider the problem of proving a trigonometric identity that children are used to doing during their schooldays. What
would they do at the beginning? They would start with one side of the identity, and attempt to apply a number of formula there to find the possible resulting derivations. But they won’t really apply all the formula there. Rather, they identify the right candidate formula that fits there, such that the other side of the identity that seems to be closer in some sense (outlook). Ultimately, when the decision regarding the selection of the formula is over, they apply it to one side (say the L.H.S) of the identity and derive the new state.

Therefore, they continue the process and go on generating new intermediate states until the R.H.S (goal) is reached. But do they always select the right candidate formula at a given state? From our experience, we know the answer is “not always”. But what would we do if we find that after generation of a few states, the resulting expression seems to be far away from the R.H.S of the identity [2].

Perhaps we would prefer to move to some old state, which is more promising, i.e., closer to the R.H.S of the identity. The above line of thinking has been realized in many intelligent search problems of AI.

**Some of these well-known search algorithms are:**

- Generate and Test
- Hill Climbing
- Heuristic Search
- Means and Ends analysis
(a) Generate and Test Approach
This approach concerns the generation of the state-space from a known starting state (root) of the problem and continues expanding the reasoning space until the goal node or the terminal state is reached [2].

In fact after generating each and every state, the generated node is compared with the known goal state. When the goal is found, the algorithm terminates. In case there exist multiple paths leading to the goal, then the path having the smallest distance from the root is preferred. The basic strategy used in this search is only generation of states and their testing for goals but it does not allow filtering of states [2].

(b) Hill Climbing Approach
Under this approach, one has to first generate a starting state and measure the total cost for reaching the goal from the given starting state. Let this cost be f. While f = a predefined utility value and the goal is not reached, new nodes are generated as children of the current node. However, in case all neighborhood nodes (states) yield an identical value of f and the goal is not included in the set of these nodes, the search algorithm is trapped at a hillock or local extreme[2]. One way to overcome this problem is to select randomly a new starting state and then continue the above search process. While proving trigonometric identities, we often use Hill Climbing, perhaps unknowingly [1, 2].
(c) Heuristic Search
Classically heuristics means rule of thumb. In heuristic search, we generally use one or more heuristic functions to determine the better candidate states among a set of legal states that could be generated from a known state [2].

The heuristic function, in other words, measures the fitness of the candidate states. The better the selection of the states, the fewer will be the number of intermediate states for reaching the goal.

However, the most difficult task in heuristic search problems is the selection of the heuristic functions. One has to select them intuitively, so that in most cases hopefully it would be able to prune the search space correctly.

(d) Means and Ends Analysis
This method of search attempts to reduce the gap between the current state and the goal state. One simple way to explore this method is to measure the distance between the current state and the goal, then apply an operator to the current state, so that the distance between the resulting state and the goal is reduced. In many mathematical theorem-proving processes, we use Means and Ends Analysis [2].

1.2.1.2 AI in 3D Games
Artificial Intelligence (AI) in games has taken the backseat in development for a long time for many reasons but the future of game is definitely going to be weighted heavily with increasingly detailed
game AI. If your game's AI is not up to the current level that game player's expectations demand then your game will feel dated and suffer for it in their opinions [3].

Game AI is not just neural networks, learning systems and complex mathematical structures, although it can be, but primarily game AI is about creating an environment and the appearance of thought from units. Game AI is behavioral, not scientific [3].

The key to understand is how to create game AI, what you want your final results to be and then building the system to provide those results. It all comes down to what the player can see; if they can't tell it's happening, then it might as well not be.

Since game AI is centered on appearance of intelligence and good gameplay, its approach is very different from that of traditional AI. A common goal today is to make the game AI more human, or at least appear so [3].

1.3 Motivation

Pathfinding in the games involves solving a planning problem with agents seeking optimal paths from a start state to a goal state. The pathfinding process involves utilizing the full state space information available to agents to find the least expensive route to the goal.

Two well-known and efficient existing pathfinding algorithms for 3D games and simulation projects are Breadth-First Search (BFS) and Dijkstra Algorithm. Among them Dijkstra is one of the most famous path finding algorithm is being used in 3d Games and Simulation project now a days. By doing literature survey, following limitations in
3D games development have been identified with existing pathfinding algorithms and they are as below[5]:

1. Breadth First Search (BFS) is a non-informed search for trees. It involves visiting the nodes closest to the root before spreading out[11,12].
2. BFS uses memory like a hog. Since we want to visit the nodes closest to the root before we visit their children, we must store the children in a queue. At each iteration, we take out a child, check to see if she is the solution. If she is, exit quickly. If not, put her children in the queue and repeat [11,12,13,14,15,16].
3. BFS must put all the children of all the nodes it traverses, the queue can get quite big, especially if the branching factor is high [11,12,13,14,15,16].
4. The major disadvantage of the Existing pathfinding algorithm like Dijkstra is the fact that it does a blind search there by consuming a lot of time waste of necessary resources.
5. Another disadvantage is that Dijkstra cannot handle negative edges. This leads to acyclic graphs and most often cannot obtain the right shortest path in the game [4,11,12,13,14,15,16].

For a more efficient and safer design of public facilities, it is important to be able to estimate the behavior of the crowd. The existing model available for the game to describe the steer behavior of the crowd usually deals with microscopic variables like the average speed or the flow. They are not designed on an individual-based model of the crowd which results in not refined steer behavior in the game system[4,11,12]. Also, there is need of the model which presents classification of the factors that influences the performance of path finding techniques [4,11].
Chapter 1: Introduction

The limitation of existing system have motivated for research work in this area.

1.4 Problem Statement
As a part of research, a new pathfinding algorithm for 3D game has been developed here to overcome the limitations of existing algorithms (BFS and Dijkstra). Using this algorithm a prototype model (third person shooter 3D game) is designed and developed for steer behavior.

1.5 Objectives
The main objective of the research is to design and develop a new pathfinding algorithm for 3D games. Which overcomes the limitation of the existing algorithms. A new model of AI system (prototype) has been designed and developed using this algorithm for steer behavior.

Features of the developed System:

1. The AiDemo game uses proposed pathfinding algorithm which is faster and more accurate Pathfinding than existing once (BFS and Dijkstra ) in Game Artificial Intelligent world. Proposed PathFinding Technique for Steer Behavior combines the pieces of information that Dijkstra’s algorithm uses (favoring vertices that are close to the starting point) and information that BFS uses (favoring vertices that are close to the goal). This research will focus on how machine learning techniques like BFS and Dijkstra Algorithms can be used to enhance an agents ability to handle pathfinding in real-time.

2. The research also includes classification of the factors that influences the performance of path finding techniques. This factors includes the dynamics of the game, the geometry of the players and the environment, the (un) predictability of movement, kinematic
and temporal restrictions, interaction rules, and real-time performance. The purpose of this classification is to help the developers to identify the complexity of the task before choosing a certain approach.

3. A focus of pedestrian’s behavior in 3D games and simulations is increased now a days in the game industry. AIDemo also contains efficient collision avoidance algorithm that can be used in simulation of crowd behavior in 3d game and simulation project. It demonstrates flock AI, Combat system AI, NPC behavior, and Crowd behavior.

Hence, the research emphasis on steer behavior and implementation of it using proposed pathfinding algorithm.

The game demo which is designed and developed for this research uses proposed pathfinding (seeking for a way from one point to another) algorithm and Steer Behavior. The demo is an ACTION game which is developed using Unity3D game engine. It is supported by wide range of platforms, the Unity3D game engine supports; the developed model also supports cross-platforms. I.e. a player with an Android device, iphone with windows desktop PC can run this game by doing minor modification in the code.

1.6 Structure of thesis
The whole research work is divided in 6 chapters. The following points give summery of each chapter.

Chapter 1 – Introduction
This Chapter covers the information about games and latest technology used in 3D game development. 3D game technology is
Chapter 1: Introduction

enormously large Technology field. It also covers overview about the technology and AI used in the 3D game development.

Chapter 2- Literature Survey & Scope of Research

This chapter discusses the work carried out in 3D game development by various developers. It also shows various areas and scope of research in Game development. Several games development, systems, articles, game development algorithms and reports have been studied for conducting review and based on those issues and challenges are identified.

Chapter 3- Study of Related Technology

There are several advancements in Game Developments which can be used to improve the overall 3D game development. Several tools and techniques have been studied for developing the AI of Game. This chapter covers the overview of tools and technologies used for implementing the AI system in games like 3D max, Unity3D game engine, java script and c#.net

Chapter 4- Overall System

It describes the newly designed and developed pathfinding algorithm for 3D game. It also explains model of AI Demo which uses proposed pathfinding algorithm and implementation of steer behavior using unity 3D game engine.

Chapter 5- System Outcome

This chapter explains how the system (AiDemo) works in detail. It explains simulation of crowd behavior in 3D games and simulation project. It also demonstrates Combat system AI, NPC behavior, flock AI, and Crowd behavior which can be used in Game development.
Chapter 6- Conclusion & Future Scope

This chapter covers conclusion of the developed system and further scope in that system.

1.7 Summary

This chapter covers overview about the AI, application areas of AI and technology used in the 3D game development.
It gives overview of existing pathfinding techniques with their limitations. It also gives idea about needs of research work in this area. In the later part of the chapter objectives, features of the proposed model are listed. The last part of the chapter gives overview of the structure of this thesis.