A Project Report on

Path Finding in Unity

Submitted in partial fulfilment of the requirements for the award of the degree of

Bachelor of Engineering

in

Computer Engineering

by

Charandeep Singh – 16102063

Meet Maisheri – 16102061

Uttam Bogati - 16102059

Under the Guidance of

Sofiya Mujawar

Department of Branch Name
A.P. Shah Institute of Technology
G.B.Road,Kasarvadavli, Thane(W), Mumbai-400615 UNIVERSITY OF MUMBAI

Academic Year 2017-2018

Approval Sheet

This Project *"Path Finding in Unity"* Submitted by Charandeep Singh (16102063), Meet Maisheri (16102061) and Uttam Bogati (16102059) is approved for the partial fulfilment of the requirement for the award of the degree of *Bachelor of Engineering* in *Computer Engineering* from *University of Mumbai*.

Sofiya Mujawar
Guide

Prof. Sachin Malave
Head Department of Computer Engineering

Place: A.P.Shah Institute of Technology, Thane

Date:

CERTIFICATE

This is to certify that the project entitled *"Path Finding In Unity"* submitted by *"Charandeep Singh" (16102063),"Meet Maisheri" (16102061),"Uttam Bogati" (16102059),* for the partial fulfillment of the requirement for award of a degree *Bachelor of Engineering* in *Branch Name.*,to the University of Mumbai,is a bonafide work carried out during academic year 2017-2018.

Sofiya Mujawar
Guide

Dr. Uttam D.Kolekar
Principal

Prof. Sachin Malave
Head Department of Computer Engineering

External Examiner(s)

1.

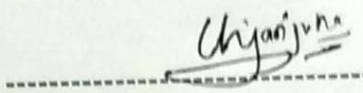
2.

Place:A.P.Shah Institute of Technology, Thane

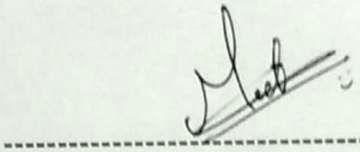
Date:

Declaration

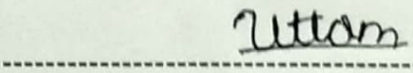
We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.



Charandeep Singh-16102063



Meet Maisheri- 16102061



Uttam Bogati-16102059

Date:

Abstract

Pathfinding is widely used in virtual environments, such as computer games. Most pathfinding types involve shortest pathfinding, which explores the fastest path, but tactical paths can also be searched for using various properties. This project involves methods for finding safe paths that maintain a balance between path length and risks, as well as a method to reduce computation time using Unity Engine, Unity is a cross-platform real-time game engine developed by Unity Technologies.

Here we're using Reinforcement method of trial and error to find the optimistic path. Secondly, we demonstrate the effectiveness of Navigation Mesh baking in Unity Engine, this process collects the render meshes and then process them to create a navigation that approximates the walkable surface of the level.

We show that Path-finding is not only able to cleanly interleave planning and execution, but it also able to do so with only minimal losses of optimality.

Contents

List of Abbreviations

NAVMESH	Navigation Mesh

Chapter 1

Introduction

Unity 3D is a cross-platform 3D game engine. That is an important problem of seeking the road in the process of game production. The main solution is path-finding algorithm in which the player finds a path to the target in game scene. In this project, by general path-finding game scene as background, we focus on the A* algorithm and "Navmesh Grid" path-finding in Unity 3D, and put forward the improvement and application of algorithm in the special game scene.

1.1.1 Subsection

Unity 3D is a comprehensive game development tool which is developed by Unity Technologies. Most games in the process of production will encounter the problem of seeking the road. How to try to find a path that from the starting point to the target point in game map? More complex games also needs to consider the file structure of the game map and the passable of target location etc. In this project, we focus on the two common path-finding algorithm in Unity 3D, and put forward the improvement and application of algorithm in the special game scene.

Chapter 2

Literature Review

LITERATURE SURVEY

SN	PAPER TITLE	AUTHOR	SUMMARY
1	Research and application of path-finding algorithm based on unity 3D	1.Zhang HE 2.Minyong Shi 3.Chunfang Li	1.A* Algorithm Theory. 2. Navmesh Grid Path-Finding. 3.Implementation Steps.
2	A Review on Algorithms For Pathfinding in Computer Games	1. Parth Mehta 2. Hetasha Shah 3. Soumya Shukla 4. Saurav Verma	1. Navigation Mesh 2. A* Algorithm 3. Future Scope

Chapter 3

A* Algorithm

A* algorithm is the most widely used in the game map, which uses the heuristic function to estimate the distance of any point to the target point, so as to reduce the search space and improve the search efficiency.

A* Algorithm Theory

A* algorithm is a heuristic search algorithm based on "Dijkstra" algorithm. Heuristic search is searching to evaluate each extension node in the state space, and select the best node's location, and then search from the start node until it finds the target node. In heuristic search, the location of the evaluation is very important, and the use of different evaluation may have different effects. The valuation function represents the estimated cost of moving from the current node to the target node, which is heuristic. In the path-finding and maze problem, we usually use the Manhattan estimation function to estimate the cost. The valuation function of the current node n is expressed as:

$$F(n)=H(n)+G(n)$$

F (n) is an evaluated function, H (n) is the heuristic value of the shortest path of any node n to the target point, G (n) is the shortest path of the start point to any node n.

It can be proved that if the evaluated function satisfies the compatibility condition that the evaluated function H (n) is less than the actual cost of the node n to the target node, and the original problem exists the optimal solution, then the A * algorithm can find the optimal path.

In the game, the A * algorithm is used to search in the map and create the corresponding node by through the game object's location to save the location information of moving objects. Before using the A * algorithm, the map should be divided to each piece as a node. Quadrilateral division is the simplest form because some terrains on the scene may be divided into a convex polygon map, four fork tree, and other shapes.

B. Implementation Steps

After the scene was built in Unity 3D, create a new class "Node" for information storage node, including the radius of the grid, world coordinates, parent node, and create a constructor for transfer value. Select the desired walking ground, and determine the number and size of the grid according to the size of the ground and the radius of the Node, Marking it where the barrier mesh layer as "Unwalkable". Determine the grid of the player by the world coordinates of the starting point of the player and declaring an open set and a closed set, and add the player's start point grid to the beginning of the collection. If the open list contains some data, circling the following steps:

1. *Set the current node to an adjacent node of the minimum F value by traversing the open list.*

2. *Remove the current node from the open set and add it to the closed set.*

3. *If the current node is the target node, end query, received by the parent node back to find a shortest path.*

4. *Traverse each adjacent node of the current node:*

    1. *If the adjacent nodes cannot be accessed or the adjacent nodes are closed in the collection, skip the adjacent node.*

    2. *Calculate the G value of the current node and the distance between the current node and the adjacent node, and then gain the new path distance.*

    3. *If the distance between the new path and the adjacent node is shorter, or the open set does not include the adjacent nodes:*

        - Reset the F value.
        - Set the parent node to the current node.
        - Add the adjacent node to the open set.

C. Algorithm Improvement and Application

A* algorithm has the shortest time in theory, but also has its disadvantages, and its spatial growth is exponential, so that further optimization is needed. It can make the table data increase gradually while finding the minimum value in all nodes by searching constantly for collection, resulting in the search process more and more slowly. For the data reading problem, some scholars put forward the A * algorithm based on the restricted area to reduce the loading of data. Because A * is a detrimental algorithm, the search may result in less than the shortest path. A simple approach is to sort the data table, simply select a node when searching. It's a good choice to use two binary heap for inserting node, because the time complexity of the insert and delete nodes is only O(log n).

The path calculated by using the A * algorithm in game scene which often appears jagged sharp and is very unreal. Therefore, it is necessary to smooth the path to achieve the purpose by changing the valuation function. When a new node changes the direction of the original path, increase the value of node generation, and gain the smooth path. The player will become an ideal point When making a path-finding plan by using the A * algorithm, there is a risk for those path planning in some cases. In a real scenario, the player can't move when meets any other obstacles because of its collision box. Change the barrier adjacent nodes' priority, when the path is generated regardless of adjacent nodes of obstacles, allowing players to complete path-finding.

Chapter 4

"Navmesh Grid" Path-Finding

Navmesh grid" path-finding is a path-finding algorithm integrated in the Unity 3D, only need o set up the map and add "Navmesh Agent" component for players. Component is the asic unit of the material form in the Unity 3D. A basic component of an object includes a nobile component, a grid component, and rendering. The position of the object can be ontrolled by moving the component, the shape of an object can be created by the grid omponent, and the material of the object can be rendered by the rendering component. Build a player who can find the way through different component.

A. Theory of "Navmesh Grid"

t's same with A * algorithm, "Navmesh" also needs to divide the map into a polygon. But he A * algorithm will work well if the polygon is regular, it isn't necessary for the latter. There are many other reasons in real game scene, such as Item properties, collision, Coordinate dimension, data storage, file size etc. The data design of item need to consider the theme of entertainment, Games nature, audience groups, operating platform, and other factors. We can detect the collision between the two objects in Unity 3D, but also can detect the collision between the specific collision, and even the use of ray proj ection detection collision. Light irradiation is the most commonly algorithm used in "Navmesh" path-finding. As shown in Figure 3, the navigation mesh is composed of arbitrary convex polygons. Gray grid represent inaccessible area, white grid represent enterable area. All polygon vertexes in the grid store in clockwise order.

B. Implementation of Path-Finding

1. *Get the target position.*
2. *Get path-finding component.*
3. *Control players' movement to the target.*

It will automatically calculate the required travel path in the path-finding process.

The above operation procedure is the most basic in the path-finding. There are still some necessary steps when we develop practical projects. Game maps are often not simple 2D graphics but the combination of 2D and 3D, so we need to select the appropriate map according to the game. In some large games, the player is often more than one, so we should plan the player's independence before making the game. The game AI is also a very important problem in the path-finding game, player will get a better experience if the virtual enemy has a more higher intelligence. The focus of the interactive product lied on the process of product display, but also in realtime interaction on system with the user. It is an important problem how to display the path in the game in a graphical way and it can be dynamically changed. Game world is a virtual place, and there are a variety of unexpected scenarios, so we need to dynamically plan based on the scene of the production process.

C. The Actual Game Development

In most of the scenes, there are some objects equipped with collision box. The player can't pass due to the collision and stuck in the collision point. It should be coupled for static objects with another judge, and it moves a certain distance to the other direction, continue to find the way. It's necessary to change dynamically the routing path, because the location information is always changing for dynamic objects. Add “Obstacle Navmesh” component, check the Carve option, the role will update the current navigation grid. In the process of moving, the path of the path can be changed dynamically.

It is more convenient to use the corner method when the shape of the grid is too long. Store the vertices of the common edge of each two adjacent polygons between the start point and the target point. Connect the start point with its nearest vertex from the start point, and then connect the next edge of the point. Determine Whether start point is connected to the second side through the obstacle area, if it's not, update the path, if it is, take two line as a path to the starting point is set to the second point and get a corner of the path.

“Navmesh” grid path-finding is a path-finding algorithm integrated in the Unity 3D. We can write scripts in the bottom layer to achieve the appropriate effect according to the different scene.

Conclusions and Future Scope

A* algorithm is the most effective method to solve the shortest path in the static grid. There is a good apply in the strategy game of the search, as well as the grid search path. Navmesh navigation grid algorithm is more used in monster path-finding of game scene and dynamic obstacle avoidance scene. In recent years, game Al development soon, a variety of Al technology is introduced to the game, such as genetic algorithm, artificial neural network computing, terrain analysis technology, the group team routing algorithm and A * algorithm. The technique solve well the game path-finding problems. With the development of graphics game, game scenes are more and more complex, the single algorithm cannot be applied to all path-finding scenario. So for different scenes, it still needs put forward the algorithm improvement and application on the path-finding.

Bibliography

1. S. M. La Valle, Planning algorithms, Cambridge University Press, 2006.

2. T. K. Whangbo, "Efficient Modified Bidirectional A* Algorithm for Optimal Route-Finding", *New Trends in Applied Artificial Intelligence Japan*, vol. 4570, pp. 344-353, June 2007.

4. M. Fu, B. Xue, "A Path Planning Algorithm Based on Dynamic Networks and Restricted Searching Area", *IEEE International Conference on*, pp. 1193-1197, August 2007.

4. X. Chen, Q. Fei, L. Wei, "A New Shortest Path Algorithm based on Heuristic Strategy", *World Congress on Intelligent Control & Automation*, pp. 2531-2536, June 2006.

5. Z.X. Li, M. Anson, G.M. Li, "A procedure for quantitatively evaluating site layout alternatives", *Construction Management & Economics UK*, vol. 19, pp. 459-467, May 2001.

6. P. Lester, "Using Binary Heaps in A * Pathfinding", [online] Available: http://www.poli-cyalmanac.org/games/binaryHeaps.htmunpublished.

7. D. Lin, "A return docking algorithm for indoor cleaning robot", *Journal of Chongqing University of Science and Technology China*, vol. 12, pp. 110-112, May 2009.

8. E. Adam, *Fundamentals of Game Design Berkeley: New Riders*, February 2010.

9. Y. Lang, "Research on collision detection method in unity", *Software Guide China*, vol. 13, pp. 24-25, July 2014.

10. H.X. Guo, "A research about interaction mechanism on Unity and HTML", *Coal Technology China*, vol. 30, pp. 228-229, September 2011.

Acknowledgement

We have great pleasure in presenting the report on **Path Finding in Unity.** We take this opportunity to express our sincere thanks towards our guide **Sofiya Mujawar** Department of Computer, APSIT thane for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards his constant encouragement, support and guidance through the development of project.

We thank **Prof. Sachin Malave** Head of Department, Computer, APSIT for his encouragement during progress meeting and providing guidelines to write this report.

We also thank the entire staff of APSIT for their invaluable help rendered during the course of this work. We wish to express our deep gratitude towards all our colleagues of APSIT for

Charandeep Singh – 16102063

Meet Maisheri - 16102061

Uttam Bogati - 16102059