Effects of Multiple Gravitational Fields on Pathfinding in a Simple Two-Dimensional Platformer

Peter Innes
BSc (Hons) Computer Games Programming
University of Derby
Derby, United Kingdom
P.Innes1@unimail.derby.ac.uk

Abstract—2D Platformers are a popular genre of video games that have seen a lack of evolution in terms of game design compared to other genres, such as 3D Platformers. Titles within the Super Mario series of games are analyzed for elements which could be improved, with focus placed upon creating more advanced game worlds and smarter artificial intelligence (AI) behavior. A simple game demo is created with the aim of creating an AI-controlled opponent able to navigate and compete against a human player in a more complex game environment. The outcome of the demo is an overall success, with further work required in order to improve the intelligence of the behavior demonstrated by the AI opponent. In addition, the search algorithm used needs to be optimized for utilization with large-scale complex game environments which contain entities that the AI can interact with.

Keywords—Super Mario, Platformer, gravity, AI, pathfinding, 2D, 3D, Nintendo, A*, game design.

I. INTRODUCTION

The 2D Platformer genre of video games is a popular genre which was first made famous through the Nintendo Entertainment System title Super Mario Bros. [1] in 1985. Numerous titles have since been released within the genre as well as Super Mario series of titles [2][3].

Despite the popularity of these titles, very little evolution of the common gameplay mechanics and features found within the genre has taken place in comparison to others, such as the 3D Platformer genre - an extension of the original genre that incorporates 3D graphics and worlds. In particular, the design of the game worlds continues to remain fairly simplistic, and the behavior demonstrated by artificially intelligent agents remains relatively simple.

This paper aims to identify the common gameplay elements of the 2D Platformer genre by examining titles within the Super Mario series of video games in order to establish which core features have remained the same since its inception, whilst comparing these developments against the innovations introduced in the 3D Platformer title Super Mario Galaxy [3]. With this knowledge, and in conjunction with research that has been performed within the area of AI Pathfinding for Non-Player Characters (NPCs), a simple game will be developed in order to establish whether it is possible for more complex NPCs to navigate a complex game environment.

II. BACKGROUND INFORMATION

Super Mario Bros. [1], the first game to be released in the popular Super Mario series of games, is largely considered to be one of the pioneers of the 2D Platformer genre and displays a number of gameplay features that are commonly used within modern-day titles within the genre.

In Super Mario Bros., the game world is represented using a horizontal side-scrolling view that follows Mario. The player is able to walk or run left and right, and also has the ability to jump, allowing them to reach platforms and other objects that are located above the ground. Gravity is applied as a constant force to the player, pulling them towards the bottom of the screen until they collide with an obstacle, at which point gravitational acceleration is reset. The game also features a number of enemies which the player must defeat by jumping on them. These enemies, however, do not feature any form of AI; instead, they typically follow some pre-set behavior, such as always moving to the left of the screen, or moving back and forth between the edges of a platform.

Analyzing modern examples of games within the genre, it could be suggested that the genre has not evolved as much in comparison to other kinds of video games. New Super Mario Bros. U [2] is one such example. Despite being released 27 years after the original title within the series, many gameplay elements remain very similar to those found in Super Mario Bros.; although additional power-ups provide players with different moves they can perform, the core mechanics still revolve around running and jumping; the game world is still presented as a primarily horizontal, side-scrolling experience, with gravity applied in a constant downwards direction. Although some new enemies have been created, many of them are still the same as those found in Super Mario Bros., and continue to display the same simplistic behavior.

This is true of the majority of 2D titles within the Super Mario series. Despite each game featuring additional gameplay features not found in its predecessors, and the continuing refinement of the gameplay formula the series is
known for, it has typically not strayed from the core elements of the genre. 3D entries within the series, however, have instead shown the opposite of this trend – they have often offered a large amount of innovation within the 3D Platformer genre.

Super Mario Galaxy [3] is one title which demonstrates this. The gameplay continues to revolve around the primary player actions of running and jumping, and enemy behaviors are still relatively simple despite the added complexity of a third dimension to the game. However, the game world is radically different, and completely alters the way in which the player must navigate across levels in order to obtain their goal.

Rather than a single terrain where gravity is always exerted as a downwards force upon the player, levels are broken up into a number of “planetoids” across which the player can traverse, ranging from large planets to small asteroids and various assorted objects. Each planetoid has a number of unique properties, such as a gravitational field which affects how high the player can jump, and surface friction that affects their movements. As gravity is exerted upon the player from a number of different sources, all with different directions and levels of force, it is possible for the player to jump between the orbits of different planetoids by affecting their own gravitational velocity in such a way that the dominant gravitational force applied is changed, causing the player to fall into the orbit of a different planetoid.

Additionally, a number of in-game objects are designed to assist the player in traversing across these planetoids. These allow the player to perform various actions, such as changing their center of gravity, providing the player with a large burst of gravitational velocity in either a pre-set or player-determined direction, or launching the player along a pre-determined path. By combining these different gameplay elements with the re-defined rules used to discretize the levels featured within the game, some unique and difficult challenges are presented to the player to overcome.

Despite this, the complexity of game design that Super Mario Galaxy offers has not been replicated within any of the 2D Super Mario titles. Additionally, these challenges remain solely player-centric, and have not been utilized to improve the behavior of NPCs.

III. LITERATURE REVIEW

In order to create more complex behaviors for NPCs such as enemies, it is important not only to utilize a suitable search algorithm to generate movement paths for them to follow, but also to discretize the game world in a suitable manner in order to allow them to traverse it successfully.

One of the most common algorithms used for pathfinding in video games is the A* Search algorithm [4]. In order to use A*, an environment will be discretized into a number of nodes (often as a series of squares, also referred to as a tile map) which contain a list of “child” nodes, or the nodes that it is connected to. Each node also contains a couple of heuristic scores which measure the distance from the node to the desired target node, as well as the current distance traversed by the algorithm upon examining the node. Utilizing these scores, the algorithm is able to determine the shortest route to the target and generate a path as a series of nodes for an NPC to follow [5].

Despite tile maps being a common method used to discretize a game world, it may not be the most suited method to a particular game being developed [4]. Within the context of creating more complex worlds for 2D Platformer games, developers may wish to create environments which comprise of non-rectangular platforms, such as circular or more complex polygonal objects, where the gravitational force applied upon the player may come from multiple sources that are not necessarily towards the bottom of the screen. Therefore, alternative methods of discretization may need to be considered and, in addition, the suitability of A* to the problem.

Research has been performed in order to better study the suitability of selected state-based search algorithms to particular kinds of grid-based node representations used for pathfinding in games. Bjornsson, Enzenberger, Holte, Schaeffer and Yap [6] performed a series of experiments to compare the A* algorithm against another search algorithm, IDA*, in a number of environments discretized into different polygonal-based grids, such as tile, octile, hex and tex-based grids. Despite their results indicating that A* is better suited to larger game environments and “realistic” game maps, IDA* is noted as performing better in tex-based and hex-based grid abstractions. These findings suggest that, depending upon the method chosen by developers to discretize their game worlds, A* may be outperformed by other algorithms, thus emphasizing the importance of selecting a suitable algorithm for state-based searching.

Another factor that must be considered is the complexity and resulting performance impact of the environment within which the chosen search algorithm is able to operate. Kumar [7] discusses the utilization of inheritance for game objects to allow for unique behavior within pathfinding based upon the interactions with an interactive object within the environment. He notes that as the number of dimensions within which the algorithm operates increase, as well as the number of interactive entities, the importance of implementing suitable optimizations for path calculation, storage and interaction with objects within the environment, also increases.

Eckerle and Roth [8] also touch upon the importance of this within the context of comparing performance of a novel search algorithm between grid-based and non-grid-based worlds. In their work, they compare their own devised algorithm against the WHCA* algorithm devised by Silver [9]. Their results show that WCHA* is not well suited to continuous non-grid-based worlds, due to a rising complexity in the data model used...
to store search results. In contrast, the OCA* algorithm devised by Eckerle and Roth [8] is much more efficient at this task, and provides a basis for the suitability of non-grid-based worlds for pathfinding.

IV. METHODOLOGY

In order to evaluate the effectiveness of AI pathfinding within a non-traditional environment in a 2D Platformer, a simple game demo was created based upon Super Mario Galaxy [3]. In it, the player is presented with either an AI or Human opponent, as chosen by the player, with the goal to score as many points as possible within a 15 second time period.

The game world consists of a number of different planetoid objects, each with a varying radius and mass which are used to calculate gravitational force exerted upon both the players and Star Bits (collectable objects which increase the player’s score when touched), following the standard formula for calculating gravitational force [10]. Gravitational acceleration is calculated as described by Wolbeck [11] and applied following the instructions provided by Kankaanpää [12] to ensure that the application of gravity is not impacted by varying frame rates. In addition, only the strongest gravitational force is applied onto the player and Star Bits in order to maintain their current position within the orbit of the planet which they are closest to.

The characters feature a number of different states that determine how gravitational and movement forces impact their current position, rotation and bounds utilized for collision detection. The two primary states, Movement and Jump, alternate depending on whether the player is in mid-air or on the surface of a planet. In mid-air, gravitational acceleration continues to build until a collision with a planet is detected, at which stage the player is positioned upon its surface based on the angle from the planet’s center to the player’s position. As the player will not always be aligned with the X and Y dimensional axes, object-oriented bounding objects are used for collision detection [13] following the separating axis theorem as described by Bittle [14].

In addition to the standard movement and jump actions available to the player, four interactive objects are present which can be used to traverse the level: Sling Stars, Sling Pods, Pull Stars and Launch Stars. Each object also has one or more associated player states which determine the handling of their update per frame. Due to their design, only one player may interact with an object at a time; they must wait for other players to finish using them before the object can be interacted with.

Sling Stars provide the player with a burst of gravitational acceleration in the direction from the center of the parent planet to the Sling Star, while Sling Pods allow adjustment of the trajectory within a 180-degree arc and alteration of the power of the shot. Due to the sheer force that can be generated by these objects and the weakening force of gravity the further the player moves away from the closest planet, the acceleration is decayed per frame to slow the player down faster until the magnitude falls below a threshold. Following this, gravitational acceleration is then re-applied as normal.

Pull Stars are objects that, upon being interacted with, will pull the interacting player towards their center, ignoring the gravitational force exerted by planets until the player releases from the Pull Star. However, they may only be interacted with as long as the player is within the Pull Star’s radius, allowing for chains of Pull Stars to be interacted with to pull the player from the orbit of one planet to another. Finally, Launch Stars guide the interacting player along a pre-set path that is calculated as a Bezier curve using control nodes, following an implementation as described by Tulleken [15]. In this state, gravitational force is completely ignored until the player reaches their destination.

The game world is discretized into a non-grid-based node graph that is organized by each planet. During the loading of the game world, each planet is split into a number of segments which are used to generate a series of path nodes around its surface. These surface nodes are then connected to the nodes on either side to create a connected ring of nodes around each planet. A number of control nodes are also generated at the corners around the planet, which are used for Bezier curve generation [15].

After this process has been repeated for each planet, more path nodes are generated and connected by ‘projecting’ a jump from each surface node, and the interaction with an interactive object if applicable. This simulates a projection of the AI through multiple time steps until either the height of a jump is reached or until a new planet is reached, at which point a new node is created at the final position of the projection and is connected to the existing path nodes.

Pull Star and Launch Star projection works slightly differently from standard jump, Sling Star and Sling Pod projections. Pull Stars will form connections with any other Pull Stars within their radius, as well as creating a path node on the surface of the closest planet. Launch Stars are projected by calculating a path from the Launch Star node’s position to the end position of the launch using the control nodes created earlier in conjunction with the A* search algorithm. The returned list of nodes is then used as control points to calculate a given position on each segment of the generated Bezier curve during the update process of the player’s Launch Star state.

In order to facilitate AI navigation of the game world, child nodes are stored within their parents along with a connection type, defined as either Movement, Jump or Interact. The AI can then use this knowledge to determine how to reach the next goal node in its path list. As all interactive objects use a common interface, the AI does not need to know the specifics
about the object, just that it can be interacted with. This allows for NPCs to make full use of interactive objects in their attempts to navigate the environment.

Finally, the AI is developed to select a target Star Bit based on one of two criteria: find the Star Bit closest to the AI’s current position; or find the closest Star Bit on the planet that currently has the most Star Bits. The AI will then attempt to find a search target using an implementation of A* as described in [5] to calculate a path to the given target, and use special AI implementations of the player states to perform the actions necessary to reach its target.

V. RESULTS

The aim of the project was to create a simple 2D Platformer game that featured a non-traditional environment for the genre for both a human and AI player to be able to navigate. This goal was achieved – the game turned into a simple competitive 2D Platformer where each player, whether a human or AI, must gather as many Star Bit objects as possible within a 150 second period. The winner is declared as the player with the most Star Bits at the end of the period. Giving the AI two different possible behaviors lead to the emergence of interesting behaviors, especially following interference with its goals by the player. This could potentially make it a challenging opponent.

The project proved to be successful in its application of gravity within the game environment. Given multiple gravitational sources of varying strengths and directions being exerted upon the player, it is still possible to allow for an experience reminiscent of traditional 2D Platformer games such as Super Mario Bros. [1], whilst at the same time adding new elements to the genre. The addition of interactive objects which utilized gravity in different ways not only allows for fun gameplay mechanics, but also helps to create more complex and interesting behaviors for NPCs to be made possible, as they can be utilized in pathfinding, as opposed to simply moving and jumping within the environment.

The game was also successful in its attempts to create a non-grid-based path node structure to be utilized for calculating movement paths for the AI, as discussed previously by Eckerle and Roth [8]. Nodes were calculated successfully around the surface of each planet for movement, and projection allowed for the ability to place nodes based on jumping being affected by the various gravitational forces being exerted. The performance cost of doing so was minimal, but the behaviors allowed helped to add greater depth to the actions that can be performed by the AI.

Despite these successes, this project only serves as a stepping stone to further improvements that could be made both to the concept from a game design point-of-view, and also to the complexity and efficiency of both the AI behavior and search algorithm utilized.

The AI behaviors utilized are very simple, and it is apparent in its selection of targets to navigate to. In attempting to find the nearest Star Bit, the AI will almost always follow the same path and continue to do so until the player interferes with its goal. Additionally, due to the random nature of Star Bits spawning, the Most Valuable Planet behavior can appear to be quite erratic, often involving repeatedly moving between two planets. Given the AI’s ability to interact with objects in the environment, further work should be undertaken to improve the quality of the AI implementation. Techniques such as neural networks could be used to have an AI agent learn how to play against a player, and allow more natural behaviors to emerge. This is a topic that has already been applied to traditional 2D Super Mario games in [16], and thus has the potential to be further expanded into this area.

Despite its common use within game development [4], play-testing revealed a number of flaws with the A* algorithm used. Notably, in searches which required traversal of a large distance, a large amount of time was taken to calculate a path to the AI’s current target. Although it was not large enough to slow down the frame rate of the game, in titles where many more complex objects are being dealt with and efficiency is of key importance, optimizations would need to be made to the performance of the search. One way would be to discretize the world further so that searches could return paths consisting of individual planets, and then perform a search from the current planet to the next. This would be far less intensive than searching the entire state space at once. Alternatively, variants of the algorithm could be used to take into account the limited amount of time available per frame to search. One example is the TBA* algorithm as detailed in [16].

VI. CONCLUSION

Many games within the 2D Platformer genre developed within recent years have continued to follow the same design ideas as those released over 20 years ago, showing little evidence of evolution within the genre [1][2]. Games within the 3D Platformer genre, however, have demonstrated further innovation, particularly regarding the design of gameplay mechanics and worlds [3], though AI behavior of NPCs remains limited. Research has been conducted into improving the abilities of AI to interact with entities in game worlds [7], and to demonstrate the efficiency of AI pathfinding within non-traditional environments [8].

This project set out to create a simple game within the 2D Platformer genre, also incorporating innovative design features from the 3D Platformer genre. Another aim was to create a game environment which features non-tile-grid Pathfinding and more complex AI behaviors than typically found within NPCs of the genre. The game was successfully created, and demonstrated the ability of AI to navigate more complex environments, with gravity impacting upon their abilities, and also the potential for more complex usage of entities within the environment in order to aid navigation.
However, a number of flaws existed within the finished implementation. AI behaviors utilized were very simple and often led to erratic behavior, particularly in response to the player’s actions. Research has been conducted into improving the AI of NPCs in this space [16], suggesting the potential for more complex AI behaviors to be developed. In addition, the A* search algorithm [5] chosen to calculate movement paths impacted performance by causing a drop in frame rate when used in consecutive time steps. Steps can be taken to improve the discretization of the world in order to allow smaller searches to be performed. In addition, investigations can be undertaken into the usage of alternative search algorithms in order to improve the speed of calculation, such as the TBA* algorithm described by [16].

ACKNOWLEDGMENTS

I would like to thank Nintendo – in particular, Super Mario creator Shigeru Miyamoto and CEO of Nintendo Satoru Iwata – for their continued efforts in providing innovation within their games, and for creating the titles in question which inspired this paper [1][2][3].

In addition, special mention also goes to Dr. Tommy Thompson and Dr. Minsi Chen for their continued support, encouragement and in developing the game demo for this project.

REFERENCES