Cave Man: A PC Game Using Dijkstra's Algorithm

Eric John D. Casuga
College of Computer Studies
Lyceum of the Philippines University
Batangas City
(+63)9157895239
ericcasuga@lpubatangas.edu.ph

Ferdinand O. Olan
College of Computer Studies
Lyceum of the Philippines University
Batangas City
(+63)9453515293
ferdinandolan@lpubatangas.edu.ph

Josiah Derrick C. De Castro
College of Computer Studies
Lyceum of the Philippines University
Batangas City
(+63)9151451244
josiahdecastro@lpubatangas.edu.ph

Abstract

Cave Man: A PC Game using Dijkstra's Algorithm is a game about survival of a man trapped in the cave in which the player is in a deep tavern and must continuously climb up and evade all the enemies that he will encounter. It was developed using Unity Game Engine and Dijkstra's Algorithm was applied in the game. The path-finding scenario was developed using features of Unity game engine unit with C# programming language. This game is two-dimensional that uses basic arrow keys to move and spacebar to jump.

Keywords: *2D Games, Dijkstra's Algorithm, Game Development, Unity3d, Unity Assets, Unity Store*

1.0 INTRODUCTION

Cave Man, using Dijkstra's Algorithm from the opponent AI's perspective is to find the path of the player whose travelling inside the cave. This game will test the skills of the player in quick decision making to survive at the deadly cave with full of skeleton enemies inside. It is a 2d platform game in which one has to reach certain doors to pass through different levels in the game. Health bar is used to keep track whether you're about to die every time the opponent hits you and there are gold coins that the player can collect and more complicated jumping paths to make it a fun and challenging game.

Dijkstra's Algorithm in accordance to *pathfinding* is tricky at first but fun way to implement in a 2D platformer game.

Luckily, we have the opportunity to use our advance software technology as of today in creating and developing a game. By applying Dijkstra's on the opponent, it will make its own path in order to chase down the player character. According to *Daniel Branicki*[2], an indie game developer from Poland, "pathfinding works well for games in which the characters can move freely along both the x- and y-axes. The problem with applying this to platformers is that movement on the y-axis is heavily restricted, due to simulated gravitational forces. Remember, it's the algorithm that uses a grid-based approach; in theory, your game and its levels don't have to".

Our chosen game engine which is called *Unity Game Engine* made our goal gain its reality because Unity were not just a game engine, but it is community-oriented game engine in which tons of tutorials, lessons and practices are available to look through and made us developed and create it. Unity is one of the easiest, non-complicated game engine out there, and it is free. This software is for everyone out there who wants to develop and create their own game.

This study developed a game that gives a 2-dimensional experience to the gamer; with the application of using Dijkstra's Algorithm in developing the game; and use the knowledge we have in C# programming for the scripts in the game, as C# is the main programming language that Unity use in doing codes for developing the game.

Objectives

1. To analyze the use of Dijkstra's algorithm.
2. To use Dijkstra's Algorithm as the path-finding method in our game.
3. To use Unity Game Engine in the development of the game.

2.0 LITERATURE REVIEW

Game Development

Game development is the process of creating and developing a game by using different programming languages and using it as a code. A person who performs this is called game developer. There are many ways of how a game developer creates his game. Anyone is capable of creating their own game, it does not take a master programmer, or a master artist. It does not require even a beginner programmer. There are enough of tools out there that you do not have to write a single line of code and you can create a finished product. If you are a 14-year old student who cannot draw to save his life, you can still make a video game. If you are a 35-year old mother of two who do not know how to program, you can still make a video game.

Most of the skills in making video games comes from following through with executing your idea, not with how you go about doing it. Again, it do not matter what software you use or what language it is coded in, it matters what you do with it.[3]

Game Development has become one of the successful industries and made game development a huge deal for talented programmers with innovative skills. It does take a lot of people in a certain company to be great, people who are embedded with imaginative skills, companies like Sony, Square Enix, Nintendo and Blizzard Entertainment are currently on their averaged revenue of 5.1 billion dollars. For hobbyist out there that made themselves successful without having a lot of man power. These people are called Indie Game Developers, who really invest a lot of time with their limited source of abilities but still had the passion to create and finish a project in which we call Indie Games.[4].

Types and kinds of video games

There are tons of types and kinds of games. It could be played in console, personal computers, smart phones and the traditional board games. It comes within types if they are 2D, 3D, or even the latest and most anticipated type nowadays, which is virtual reality.[1]

Console games and computer games have the power to support and run high-end games which is really particularly played by most of the gamers. [5]Mobile games, which usually come in 2D, but as of today smart phones are ruling over you can actually play most of the advanced graphic games that PCs and consoles do.[13]

Game Genres

The broadest definition of what a game is. We believe you can file almost any game under each of the below genres, while also applying appropriate subgenres to its classification. In some uncommon occurrences, genres may cross over, but they generally keep to themselves.

.

- ✓ *Action:* The broadest genre out there, is any game where fighting enemies is the main thing that the player does in a game, or navigating your way around the environment to reach a goal, where you retain control on a more personal level. Anything from the Brawler subgenre, like God of War or Streets of Rage, or the Shooter subgenre, like Call of Duty or Bioshock, to the platformer subgenre, like Mario and Rayman.

- ✓ *Strategy:* Games where you, as the name implies, primarily strategize on a more broad level. Games like DOTA and Starcraft.

- ✓ *Racing:* As self-definitional as it sounds, any game where racing is the primary objective. Games like Mario Kart and Gran Turismo.

- ✓ *Fighting:* This one is a bit weird. One would think, without any prior knowledge of gaming, that this would fall under the "Action" genre, but it is such a well-defined consistent style that ends up being its own genre. Something like Street Fighter, Tekken, or even Smash Bros.

- ✓ *Puzzle:* Any game where you can use problem-solving or quick thinking to play. Tetris and Bejewled are the more fast-paced kind, while something like Portal requires more drawn-out thought.

- ✓ *Rhythm:* Games that primarily use rhythm and music as the driving factors. Games like Guitar Hero and Rhythm Heaven.

✓ *Sports:* Game simulations of sports that are played in real life. Titles like Madden and FIFA.

✓ *Simulation:* Simulates events you can do in real life (that are non-sport), like Microsoft Flight Simulator, or some of the more silly ones like Surgeon Simulator and Goat Simulator. Can often contain games that have no real objectives. Could possibly be called "toys" in this case.

Gameplay Elements

Gameplay elements are part of what makes a game unique, and define how you would play it. From these gameplay elements, we can find common traits and start to group things by sub-genre.

Sub-genres

As stated above, anything within the umbrella of a genre that differentiates itself from other games in said genre. It is important to note that a game can consists of two or more different subgenres.

Common subgenres: Brawler, Shooter, **RPG**, **Stealth**, Platformer, **Sandbox**, Real-Time Strategy, Turn-Base strategy, **Adventure**, **Builder**.[7]

Use of Algorithm in Gaming

The fundamentals of algorithms (like big-O notation) are not that hard, and overlap heavily with knowing about common data structures like binary search trees and graphs. That said, something like Sudoku or Monument Valley is probably fine with naive brute force solutions because the problem size is so small; the parts of a game that really need to be super-efficient are normally hidden inside the "engine".[6]

There are too many types of computer games; each type has its own knowledge requirements, so one cannot exhaustively explain everything involved. A person will need to have a basic level of aptitude to understand the use of Algorithm. It is expected that a person with basic level of aptitude will not ask the question "is it necessary to learn algorithms" because it would have been affirmed by your own learning experience.[10]

Dijkstra’s Algorithm

Dijkstra's *(dīk-stra)* shortest path algorithm is an acquisitive algorithm that operatively seeks the quickest paths in a graph.

Numerous more issues than you might think at first can be cast as shortest way problems, making this algorithm a capable and common instrument. For example, Dijkstra’s algorithm could be a great way to execute a service like MapQuest that finds the most limited way to drive between two focuses on the map. It can, moreover, be utilized to solve issues like network routing, where the objective is to find the shortest way for information packets to go through a switching network. It is furthermore utilized in more general search algorithms for an assortment of issues extending from automated circuit format to speech recognition.

"Dijkstra's algorithm is an algorithm that will determine the best route to take, given a number of vertices (nodes) and edges (node paths). So, if we have a graph, if we follow Dijkstra's algorithm we can efficiently figure out the shortest route no matter how large the graph is."

The Use of C# in the Game

By characterizing an information foe weighted and directed charts, we are able to express algorithms free from the usage of the charts themselves. In a weighted chart, each of its edges has a positive weight that we are able to think of as the disposition one must go to when tracing going along that edge.[7]

```
(* A signature for directed graphs. The signature is
 * simplified by not explicitly representing edges as
 * type. *)
signature WGRAPH = sig
  type graph  (* A directed graph comprising a set of
             * vertices and directed edges with nonnegative
             * weights. *)
  type vertex (* A vertex, or node, of the graph *)

  (* Whether two vertices are the same vertex. *)
  val eq: vertex*vertex->bool
  (* All vertices in the graph, without any duplicates.
   * Run time: O(|V|). *)
  val vertices: graph->vertex list
  (* outgoing(v) is a list of pairs (v_i,w_i), one for each
   * edge leaving the vertex v. For each index i, the
   * corresponding edge leaves v and goes to v_i, and
   * has weight w_i.
   * Run time is linear in the length of the result. *)
  val outgoing: vertex->(vertex*int) list
end
```

There are a few limitations on the running time of certain operations in this detail. Vitally, we expect that given a vertex, we are able to navigate the outgoing edges in constant time per edge. Some graph usage do not have these properties, but we will effortlessly write a nearly trifling usage that does:

```
structure Graph : WGRAPH = struct
  (* Note: vertex must contain a ref to allow graphs
   * containing cycles to be built and to give vertices
   * a notion of unique identity (ref identity).
   * The type vertex must be a datatype to permit it to
   * be defined recursively. *)
  datatype vertex = V of (vertex*int) list ref
  type graph = vertex list

  fun eq(V(v1), V(v2)) = (v1 = v2)
  fun vertices(g) = g
  fun outgoing(V(lr)) = !lr
end
```

"A **path** through the graph is a sequence (v_1, ..., v_n) such that the graph contains an edge e_1 going from v_1 to v_2, an edge e_2 going from v_2 to v_3, and so on. That is, all the edges must be traversed in the forward direction. The **length** of a path is the sum of the weights along these edges e_1,..., e_{n-1}. We call this property "length" even though for some graphs it may represent some other quantity: for example, money or time." [10]

Implementation of the Algorithm

Once more, let us begin to characterize what precisely the issue is. Take this graph, for instance. Look at the Figure1.[9]

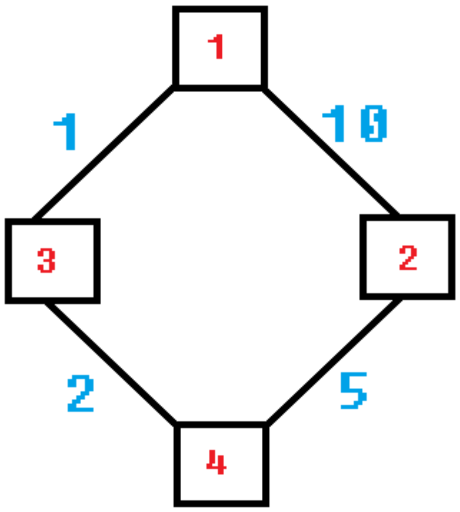


Figure 1: Node Graph

For the purposes of this study, the blue circles represent "nodes" or "vertices" and the dark lines are "edges" or "node paths". Each edge incorporates a cost related with it. For this picture, the number in each node is essentially a name for the node, not the individual node cost.[11]

Our issue at hand is to discover the most fetched productive course from Node 1 to Node 4. The numbers on the node paths portray the "cost" of coursing through the nodes. The quickest path from Node 1 to Node 4 is to take Node 1 to Node 3 to Node 4, as that is the path where cost is least sustained.

To be exact, the price to go from Node 1 to Node 3 is (2), plus the price of Node 3 to Node 4 (5) is 7 (2 + 5).

Now, we can see that the alternative (Node 1 to Node 2 to Node 4) is pricey (it costs 11, versus our 7).

An critical note - greedy algorithms are not really successful in this situation. A greedy algorithm would fundamentally find the least costly local costs as it navigates the graph with the expectation that it would be universally ideal when it is done. Meaning, an eager algorithm would basically just take the first low value it sees. In this case, the lower value is 1 but the next value is 10. If we just simply apply a greedy algorithm, we will end up taking the costlier route from Node 1 to Node 4.

Determining the best path to take with this graph is quite easy for us to do mentally and as if you can add small numbers, you can point out the best path to take. The objective is to translate the procedures we undertake in our mind to the steps a computer can conform to.[8]

"Dijkstra's algorithm provides for us the shortest path from Node A to Node B."

This prominent concept is basically how Google maps furnish you with directions. There are a multiple of vertices and edges, and when you ask for directions you customarily want the shortest, quickest and/or the least expensive route to and from your destinations.

From the foregoing, how does this apply to AI? Well, the interrelationship is quite strong. In a 2D grid or tile based map, there are many nodes (or tiles) and each tile can have a value associated with it.

You can organize your tiles in order for each tile to have a node path value associated with it, so when you put a non-player character (NPC) in the map you can use Dijkstra's algorithm to compute the quickest path for the NPC to take to any tile in your map.[14]

Why Dijkstra’s algorithm was used?

A* vs. Dijkstra

If you are familiar with the A* algorithm, you might notice that adding the heuristic to Dijkstra's algorithm gives you something very similar. Basically, Dijkstra's algorithm with a heuristic is equivalent to A* except for a couple of technical facts:

- A* will generally utilize less memory
- You must know the whole search space to utilize Dijkstra's algorithm

For individuals that are composing tile based games, it is exceedingly prescribed to utilize Dijkstra's algorithm since it is less demanding to code and will be about as productive as A*.

3.0 METHODS

Game type to be used

As we are aiming to develop a 2D platformer game in which a player can enjoy the thrills in surviving inside a deadly cave, this is a type of game where you will roam around, attack enemies if necessary, dodge them if you can in order to keep your character alive. Different levels were also developed within the game with basic controls of WASD and Q for attack while spacebar for jumping.

Sound/Music

Without music, we cannot justify the feelings we must endow within the user who will play our game. This game is composed of thrilling background music with a bit of special sound effects for actions like attacking, jumping, enemy’s attacking and claiming of coins. All the sounds and music in this game were mixed by the help of TwistedWave Online.

Visual Designs

All the sprites and backgrounds that were used in this game were personally designed by the help of Photoshop Online and MS Paint.

Gameplay

A nocturnal man from way past the medieval age was trapped roaming in the forest in search for gold treasures for his fortress. But when the storm came, out of desperation, he pursued to go inside the dark mysterious cave. And by his grit to collect, he goes want on and on deeper being unconscious about his surroundings. Upon his way in, the man slipped and fell deep. Without knowing the way back to the top, he must roam around and climb his way up to survive. But worse come to worst that he discovered that he was not the only one in the cave, but deadly skeleton enemies were waiting for him. He must dodge and fight his way back to the top in order to get back to his fortress.

4.0 DISCUSSION

The results achieved in creating this game have met beyond our expectations so far. Speaking for its physics in game, there are many lot more to polish. But we think the overall design was good.

Although we only managed to create three levels in this game, giving us more time to patch it up will make this even a greater game.

Flowchart

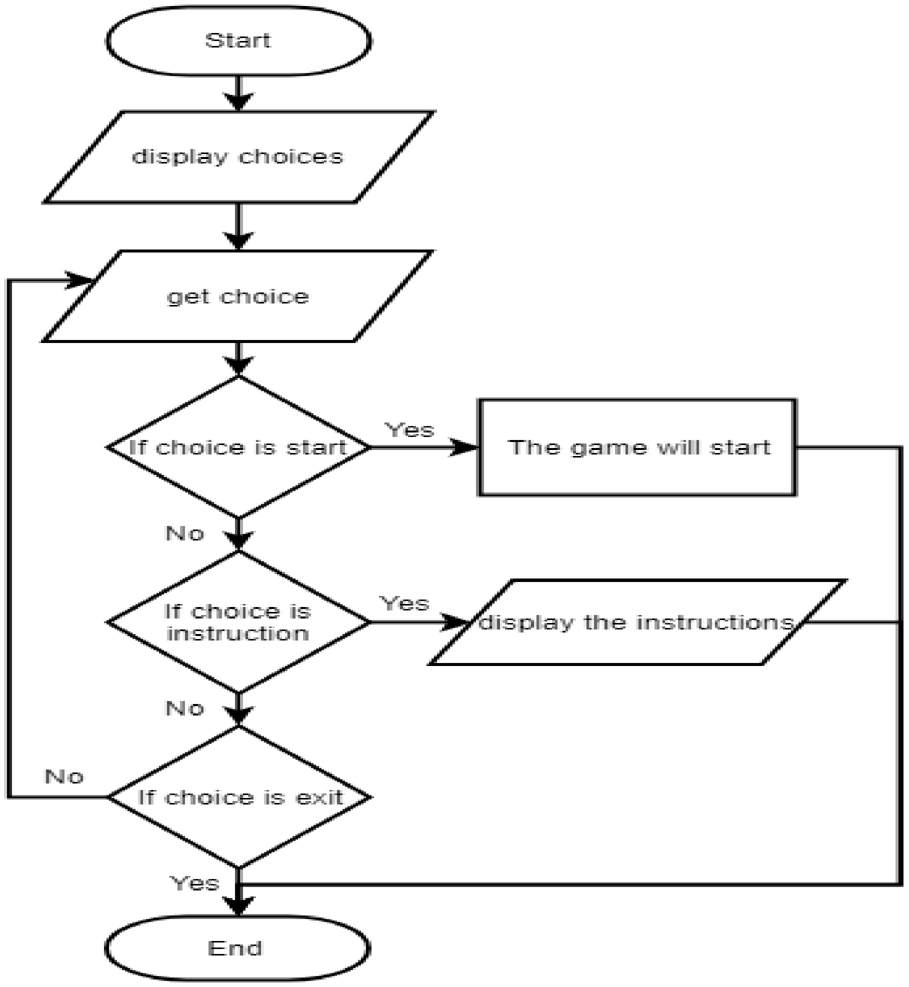


Figure 2: Main Menu

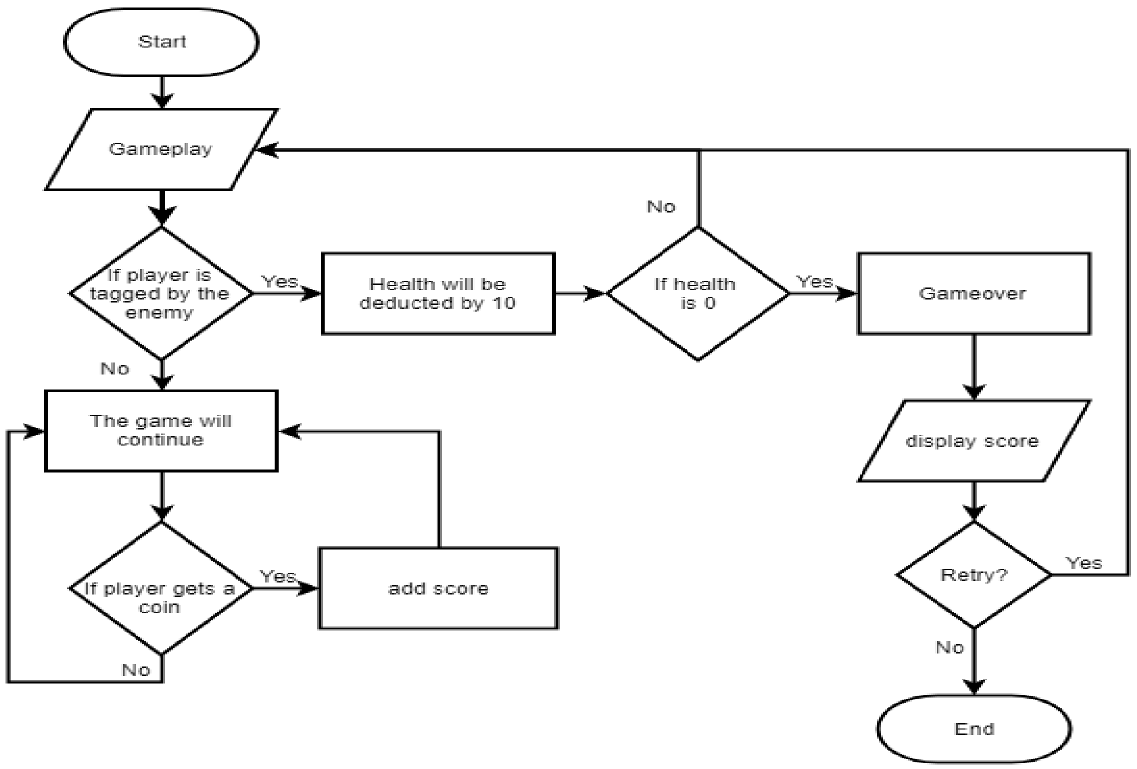


Figure 3: Game Flow

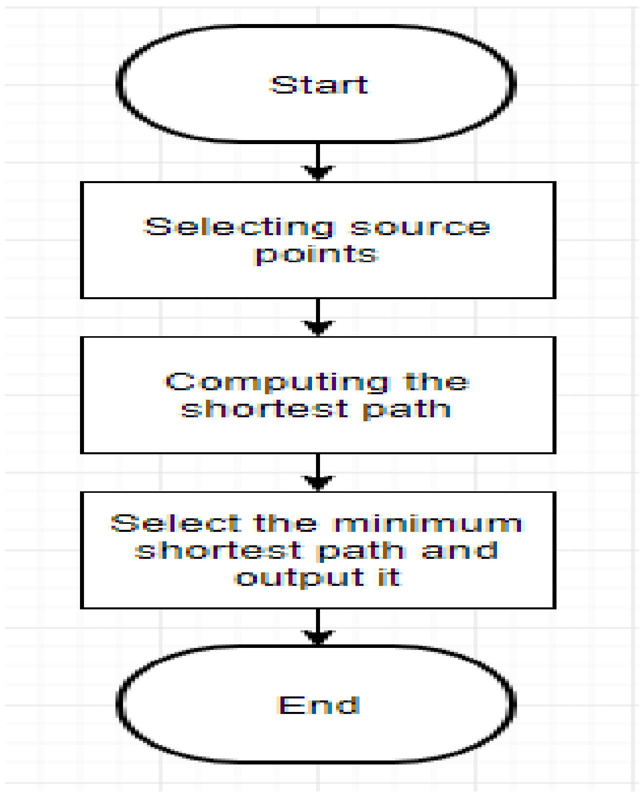
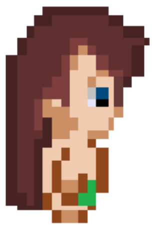

Start
Selecting source points
Computing the shortest path
Select the minimum shortest path and output it
End

Figure 4: Dijkstra’s Algorithm

Screenshots

Figure 5: Logo


Player

Enemy

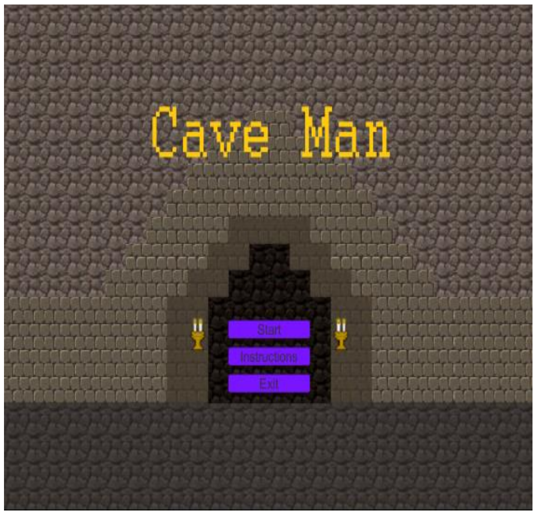


Figure 6: Main menu
This is main menu contains Start, Instructions and Exit

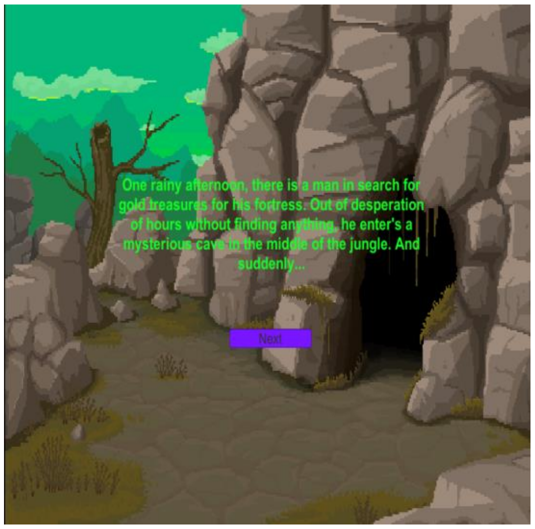

One rainy afternoon, there is a man in search for gold treasures for his fortress. Out of desperation of hours without finding anything, he enter's a mysterious cave in the middle of the jungle. And suddenly...

Figure 7: Pre-game scene
This is the prologue scene where the motivation of the character explains

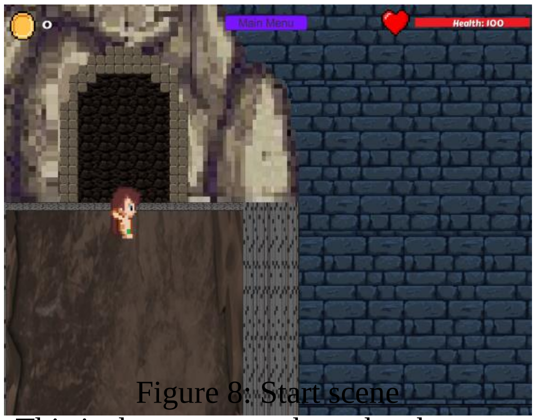


Figure 8: Start scene
This is the moment where the character fall down the cave

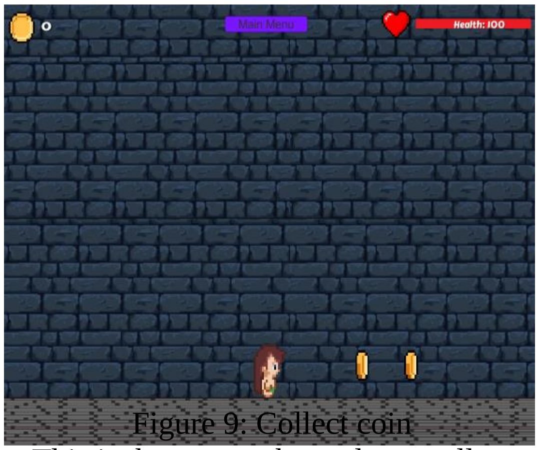


Figure 9: Collect coin
This is the scene where player collect coins

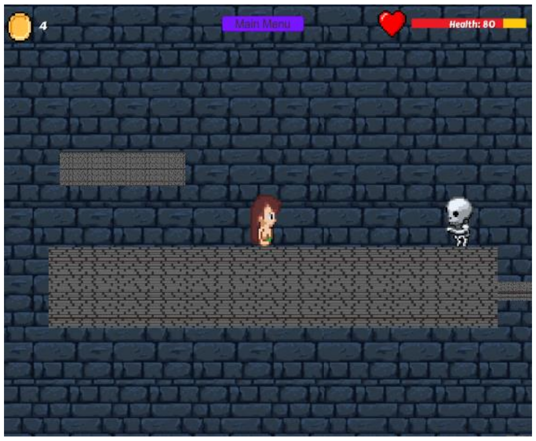


Figure 10: Enemy encounter
This is the scene where character encounter enemy

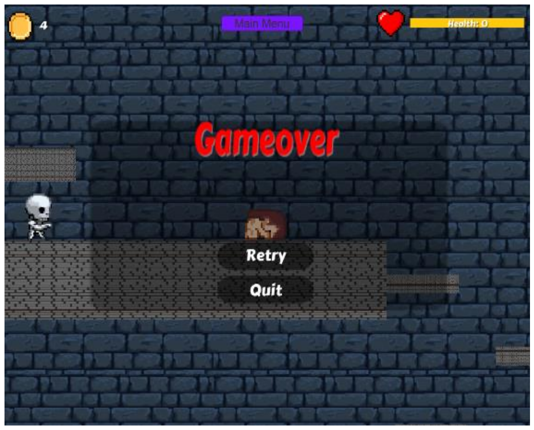
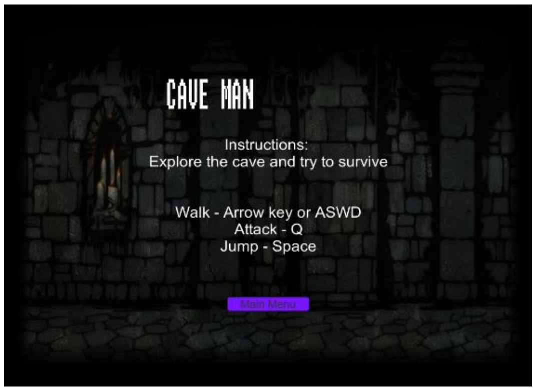


Figure 11: Gameover
This is the gameover scene where player lose.



Instructions:
Explore the cave and try to survive

Walk - Arrow key or ASWD
Attack - Q
Jump - Space

Figure 12: Instruction menu
This is the instruction menu where explain how to play the game.

5.0 CONCLUSIONS AND RECOMMENDATIONS

CONCLUSIONS

Based on this game research, the researchers came up with the following conclusions:

1. Upon the development of *Cave Man*, the researchers were able to develop a 2D-platformer game by applying to the character the use of *Dijkstra's Algorithm.*

2. The researchers were able to make a thrilling game application by applying Dijkstra's Algorithm as a pathfinding algorithm for this game. The researchers adapt the pathfinding method to the enemy character for it to track down the main character.

3. The use of Unity3D engine application enables the researchers to create a 2D-platformer game. Unity3D made use of C# programming for the scripts on the game, compelling with the assets that the researchers used.

RECOMMENDATIONS

For every aspiring game developers like us we the researchers highly recommend that if you really want to develop a game you just have to do it. Never doubt your ability if you think and feel that you really have it. As time is the essence of creating this project, more polishing should be done on UI, colliders, and enemies and scoring.

If you want to improve this game, the researchers recommend that you need to add more enemies, more power ups, more levels, change the background music to more thrilling music, exciting and more challenging, add more story to it because that is what the player wants to play. If you add more stories to it, the player will be excited to play the game.

REFERENCES

[1] Avery, P., Togelius, J., Alistar, E.: Computational Intelligence and Tower Defence Games. In: IEEE Congress on Evolutionary Comp. (CEC), pp. 1084–1091 (2014)

[2] Branicki, David. Researh specialist about game development from Poland (2015) https://gamedevelopment.tutsplus.com/tutorials/how-to-adapt-a-pathfinding-to-a- 2d-grid-based-platformer-theory--cms-24662

[3] Bwob of Paper Dino Software, Game Development company (2016) https://www.reddit.com/r/gamedev/comments/43m2sy/important_or_useful_algorithms_and_math_concepts/

[4] C. Fairclough, M. Fagan, B. Mac Namee, and P. Cunningham, "Research directions for ai in computer games," Trinity College Dublin, Department of Computer Science, Tech. Rep., Oct.Ding, W. and Marchionini, G. 2015. A Study on Video Browsing Strategies. Technical Report. University of Maryland at College Park.

[5] J. P. Gee, Videogames are good for your soul, Champaign, IL:Common Ground, 2014.

[6] K. Erenli, "The impact of gamification: A recommendation of scenarios for education", 15th International Conference on Interactive Collaborative Learning (ICL), pp. 1-8, 2014.

[7] Lance_Drake. Reddit community member (2015) https://www.reddit.com/r/truegaming/comments/3kk715/my_definition_of_video_game_genres_in_the/

[8] P. Lester, "Using Binary Heaps in Dijkstra's Pathfinding" 2016, [online] Available: http://www.poli-cyalmanac.org/games/binaryHeaps.htmunpublished.'

[9] Rouse, Margaret, Game enthusiast and blogger (2015) http://whatis.techtarget.com/definition/gaming

[10] S. Deterding, D. Dixon, R. Khaled, L. Nacke, "From game design elements to gamefulness: defining “gamification", 15th International Academic MindTrek Conference: Envisioning Future Media Environments (MindTrek '11). ACM, 9–15, 2015.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein et al., Introduction to algorithms. MIT press Cambridge, 2015, vol. 2.

[12] T. K. Whangbo 2014, "Efficient Modified Bidirectional Dijkstra’s Algorithm for Optimal Route-Finding", New

[13] WATT, Alan & POLICARPO, Fabio. "3D Games, Vol. 2: Animation and Advanced Real-Time Rendering". Harlow, England: Addison-Wesley,2015.

[14] Woodcock, Steve. "Game AI: The State of the Industry 2015-2016". Game Developer Magazine, Vol. 9, No. 7, pp. 26-31, July 2015.

[15] W. Suen, J. Hughes, M. Russell, H. Lee, A. Carr, V. Parker, S. Rao, From Role Play to Real Play: Teaching Effective Role-Playing Facilitation Skills, 2014, [online] Available: https://www.mededportal.org/publication/8603