Proceedings of the Sixth International Conference on Trends in Electronics and Informatics (ICOEI 2022)
IEEE Xplore Part Number: CFP22J32-ART; ISBN: 978-1-6654-8328-5

Development of a 2D Game using Artificial Intelligence in Unity

R. Mahizhini Sindhu
Information Technology
St. Joseph's College of Engineering
Chennai, India
mahizhinisindhu2000@gmail.com

L. Sherly Puspha Annabel
Information Technology
St. Joseph's College of Engineering
Chennai, India
sherlyannabell@stjosephs.ac.in

G. Monisha
Information Technology
St. Joseph's College of Engineering
Chennai, India
monisha090301@gmail.com

***Abstract*—The game industry comes to life like never before thanks to ever-evolving technology, as new games are released every day and promising remakes are released every year. The gaming industry is getting more and more vibrant as an interactive and entertaining medium for a wide range of people of all ages and backgrounds, as its revenue rises. The main goal of our 2D video pixel art game named 'Disappear' is to create awareness of the 'Mental Health' issues. When the game requires a particular character, the Artificial Intelligence needs to mimic as that Non-Playable Character. It helps the player more engaged in the game space while exploring the game. While fighting against the mob, player will be facing difficulty based on the enemy AI. The AI will have their own set of rules and their own roles to play, which are already enforced in their decision making. By engaging the players in the game, the awareness of depression will be conveyed subtle manner.**

Keywords—Game Development, A.I. agent, Non-Playable Characters (NPCs), Decision making, Goal-oriented behavior, Action-Platform game

I. INTRODUCTION

Once before 3D games, 2D games are the most popular video game of all time. It was in the 1980s and mid-1990s 2D platform games are started to bloom all over the places. Platform games used to be played on a screen, with the player having to pass all the various barriers. The scrolling level games are beginning to emerge in the 2D platform games with the release of "Super Mario Bros.,", in which Mario scrolled from one side to the other side while viewing player in the screen. The player jumps from platform to platform, wielding weapons to battle foes. Outside of the games industry, platform games are the popular games that is known by many, which is why most of the games based on movies, stories, television programs, and comic books tend to be platform games. Artificial Intelligence for Games (Game AI) is primarily concerned with determining which actions an entity should take in context of given situation. This is referred as commanding 'intelligent agents' in traditional AI literature, where the agent is usually a game character – but could be anything. In each scenario, an object must study its surroundings make decisions based on those observations, then act on those decisions. Sense - The agent senses things in their environment that may have an impact on their conduct (e.g. threats nearby). Think – The agent decides what to do in reaction to the situation (e.g. when it should attack). Act - The agent takes action to carry out the earlier decision (e.g. move towards the target and attack). Because the circumstances will change as a result of the characters' actions, the cycle must be repeated with the new data.

Because so much of this information is inherent to the simulation, 2D games are uncommon in that they don't usually require a complicated mechanism to extract it. There's no need to use image recognition algorithms to figure out if there's an adversary ahead; the game already knows there's one and can use that information to make decisions. As a result, the 'sense' element of the cycle is frequently much simpler, while the 'think' and 'act' implementations are often more sophisticated. The context of the paper is laid out as follows: The literature review and Finite State Machine AI in Video Game are explained in Sections II and III. The Game with Finite State Machine AI and conclusion are explained in Sections IV and V.

II. LITERATURE REVIEW

Pichit Promsutipong and Vishnu Kotrajaras, wanted to create a player AI that can play in the place of human to test a game. Each time the Player AI performs a game play the evaluation will be determined by the generated enemies [1]. Apostolos Melinones and Ioannis Plas, studied two 3D based game on elementary AI adaptation. The initial game is called "Danny's Nightmare" which has the very basic AI using simple Finite State Machines (FSM). The second one, "Escape from #98" uses a straight forward case-based adaptive AI, that is more useful in intelligent applications, in which when the player escaped from the enemy AI, it will start patrolling on the place where it lost the player, thus adapting to the place and making decision based on the situation [2].

Antonio Sag and his partner Tihomir Orehovacki developed a game which uses Construct 2 game engine that is mainly based on drop and drag method which makes the process of the game making fairly easy. Small or big game making requires lots of knowledge and experience which explains the usage of Game engine to make things less complicated [3]. FSM is a finite number of states in which by adding more behavior it gives a chance to build one scenario in the many situations that can happen. This analysis of the Multi Finite State Machine with FSM is compared [4]. Ivan Bravi and his colleagues Diego Perez-Liebana, Simon M. Lucas and Jialin Liu made an examination in General Video Game for the basic game agents and MCTS-based agent which helps to shape the basic agent to be more complex [5].

There are many algorithms for Game AI, but for different kind of environment there'll be different kinds of AI algorithm [6-7]. Jie Hu with his partners Wang gen Wan and Xiaoqing Yu created an AI that'll use A* algorithm to find the shortest distance between two object in real time. This helps in the games that is not previously aligned the enemy or the player in the same position in the Unity 3D [8]. While the path finding AI which uses traditional A* algorithm to find the short distance between a two game object, it may take some time to change that Regions Discovery Algorithm divides the maps into small region which helps it easy to find distance between game object in a top down game [9]. Like top down game, platform games never had any A* algorithm

2022 6th International Conference on Trends in Electronics and Informatics (ICOEI) | 978-1-6654-8328-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICOEI53556.2022.9776750

to work for it. Umar Affandi Shahrin Iskander, Norizon Mat Diah and Marina Ismail used grids for the platform games to use A* algorithm on the platform games [10].

Most of the games don't require any common AI technology in it, there present a large difference between the Game AI and the Common AI. But one can make use of presented AI technologies in the game. Geoff Skinner and Toby Walmsley use Deep learning neural networks to train game bots [11]. This types of AI mostly used in a game with the same enemy but with different difficulties. Pacman is one such game that has common enemies which known as ghost. This game uses Artificial Neural Network (ANN) method to compute the resources, which makes the game simpler [12].

Rebecca Brown and Curry Guinn created an AI agent that can adapt user's strategy. It starts a simple AI with basic function after playing for a while with the real players it adapts the techniques [13]. While having an adaptation enemy AI, the one thing that lack is a Team-mate AI which can understand their human player and make right decision based off their game play. Aswin Thomas Abraham and his colleague Kevin McGee made effective team-player using AI [14]. There are many types of games that can be tested using AI, one of which is "Tower Defence" game, which requires the adaptive algorithm to determine what kind of towers and where to place them. Paul A. Rummell created a simplified adaptive algorithm for Tower Defence testin with one type of tower, which can possibly extend for more [15].

Leonardo Jeanny Pragantha and Darius Andana Haris created a 2D Top-Down game in which the player can move and attack the enemies that are approaching him. The enemy spawn is randomized and for each enemy death the enemy level is increased. This game uses FSM AI but it is purely used for entertainment purpose [16]. Renza Andrea, Sefty Wijayanti and Nursobah made a game in 3D with a small story that makes the player to go through a forest and make wise choice to survive the forest. They use the FSM AI for the player and test the game [17].

From the literature survey it is understood that many of the existing works describe only the methodologies without implementation. In addition, social issues like mental health issues, are not addressed in many of the existing works. These works also concentrate more on entertainment purposes. In order to address these issues, we developed a game named Disappear that create awareness about mental health issues which also entertains the players with its gameplay.

III. FINITE STATE MACHINE AI IN VIDEO GAMES

The purpose of game AI is to give human players the feel of playing with other human who can also make a reasonable decision [2]. For this to be accomplished, most of the Game AI uses a working game AI strategies like FSM, decision trees, path finding and scripting which are used on nowadays games. In gaming, AI refers to adaptable and responsive video game experiences enabled by non-playable characters responding creatively as if they were commanded by a human game player. A FSM AI is used to create more than one state and decides which state should be used on what occasion. The conditions that decide when the state should update can also be defined by a FSM. The states can also be triggered to change when a certain amount of time has passed. This trick is applied when the animation of said thing needs to change its animation due to certain time passed. The state machine's actions are determined by its current state. FSMs are extremely easy to use and may be used to provide all player controls and handling. FSMs are a set of design patterns that are frequently used in developing games.

FSMs are a very useful AI in game development that creates a work flow for the enemies by aligning their actions. FSM's function can be simply explained by a graph in which the nodes are the states and the edges are the transitions that changes a state to another state. A label on each edge indicates when the transition should occur. Their easy to understand nature and very simple application makes it still known all around the game development world. With a very basic knowledge anyone can find errors.

FSM contains one of many hypothetical actions that can be performed by the NPC that are called states. It could be the patrolling, fighting, getting hit, idle, sleeping even dying action. Using state machine, it gives an idea when to do those said actions. When a player comes near an opponent it doesn't make any sense for it to still patrolling. It needs to make a decision whether to attack or not. While attacking, it needs to perform the fighting action not the same patrolling action. To transit from one action to another action a decision is made by the NPC with the help of states, triggers and programs which are contained in the FSM. Decisions and what action needs to be performed will be determined by program while the state is triggered to transit in to the determined actions. The FSM is like a diagram which helps the developer to visualize the whole actions that can be performed by the NPCs.

IV. GAME WITH FINITE STATE MACHINE ARTIFICIAL INTELLIGENCE

Ia 2D game named "Disappear" is developed using Unity engine. This game illustrates the FSM AI in which all NPCs' behaviors are already predicted and pre-programmed to deal with them in any situation.

A. Disappear Gameflow

The Game flow follows as below: the Adventurer will wake up in an area filled with lilies. He would start wandering himself into the area that he woke and soon find himself in familiar and yet unfamiliar forest. As a trained hunter he knows how to hold his handy sword and his bow and arrows. He has to pass through the stages where each stage has different environments and its flora. The enemies are present in this varies stages blocking his path to move forward. This game totally has eight stages, with its different kinds of enemies. In each stage there will be different kinds of difficulties which will challenge the player.

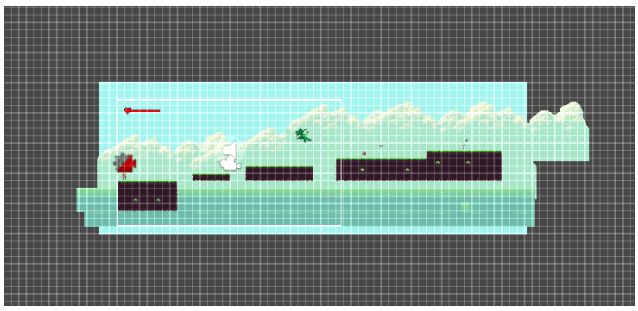

Fig. 1. Disappear Game 2D Elements placed at level's space

The Fig.1. shows the space where the game elements are put together for the game to be played. The elements are put

in a grid cells which helps to align the Game objects in the respective position. Here a game component from Unity known as grid helps in the alignment of game objects like tiles (platform where the player runs, background etc.) in their respective position. The grid used on here is the default square grid, which is obvious choice for most of the 2D games. One can drag and drop the tiles on the grid. If the size of a tile and the grid cell is the same they fit perfectly and give a desirable shape for the environment. If not, they might overlap with each other which also give a chaotic design.

B. Finite State Machine AI Design

The FSM AI performs the actions that are already predicted to happen and creates the functions for the enemy to handle such situation. In this game "Disappear" when the game starts, the enemy starts patrolling a certain area while checking the surrounding to find the adventurer. When it detects the adventurer it stops on its tracks and starts heading towards him or if it's a ranged enemy it looks at the direction of the adventurer. When it reached the perfect distance it starts hitting/ shooting the adventurer. If the adventurer escaped the area, it starts patrolling the area again, else the adventurer or the enemy will fight till death. Fig.2. shows the execution flow of the game AI. Here the FSM diagram shows the brain of the Disappear game enemy AI.

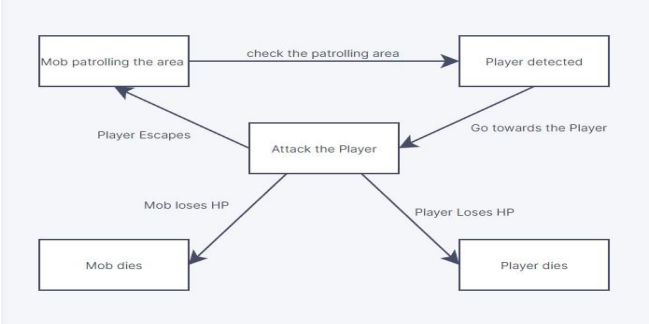

Fig. 2. Disappear Game AI Finite State Machine Diagram

The above diagram is represented using state transition table which is illustrated in Table 1.

TABLE I. STATE TRANSITION TABLE

Current State	Event	Output State
Mob Patrol	Check Patrol area	Detect Player
Detect Player	Go towards player	Attack Player
Attack Player	Player Escapes	Mob Patrol
Attack Player	Player loses HP	Player Dies
Attack Player	Mob loses HP	Mob Dies

C. 2D Model Sprite Sheets

Adventurer is a hunter who accidently finds himself in place that feels real and unreal at the same time. He wants to pass through this place that he woke up and find his way back. His sprite is presented as a man in his twenties with a red scarf in his hunting outfit.

Scythe Skeleton is an enemy that just wanders in particular area, in which the adventurer needs to defeat it to pass through. This is a Skeleton which is slightly bigger than the adventurer and wields a Scythe. The Scythe Skeleton usually patrols an area, while doing so if it detects the player with speed it approaches the player and tries to attack him. If the player escapes the place the skeleton will go on and start patrolling again.

Morningstar Skeleton is an enemy that has the same mechanics as the Scythe Skeleton with the patrolling and detection. But this skeleton uses a different weapon called Morningstar that can hit enemy and deal more damage than the Scythe.

Dragon is a flying enemy that stations in a high place and moves only up and down keeping watch over its area. When the player tries to pass through that place the dragon starts to spit fire balls which will be directed to the direction of the adventurer. The dragon is hard to defeat due to flying, the adventurer can only defeat it using his ranged weapon (bow and arrow) or the jumping slash attack. It can also deal large damage.

Mage Skeleton is also a skeleton enemy who can use magic to attack its enemy from range. He also shoots fire balls like the dragon from his staff by summoning. He has more health and attack power than any enemies.

D. Adventurer's Design

a) Adventurer Creation: This part of section gives an idea how the adventurer has been created in Aseprite. It is an image editor that is used to mostly create pixel art animation and drawings. Using the colour palette in the folder or by choosing our own, one can draw, edit and create frames for animation. The artwork can also get from other artists who shared their work for educational and non-commercial purpose. The platform game requires the adventurer to have his Rigidbody2D for the Unity's physics in this case gravity to apply on him and freeze the Z axis for him not to rotate in a 3D plain. Collider is another component which added to make him collide with other game objects. The sprite sheets for adventurers are idle, patrol, fall, jump, 3 sword attacks, jump sword attack, range attack, jump range attack and death.

b) Adventurer's Movement: This part of section gives an idea of how the controlling of adventurer movement was created. For the adventurer to stay idle move left and right he needs animation that is made for him.

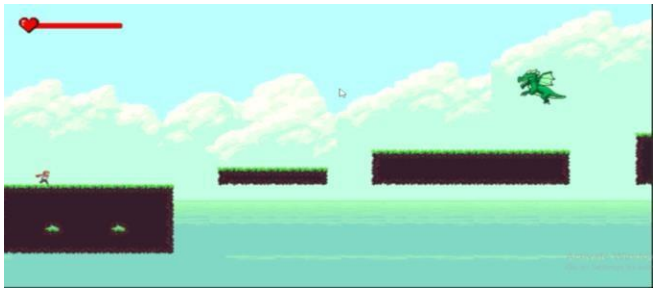

Fig. 3. Disappear Game Player's Running Movement

Unity has a built in function to get the movement input which is enough to get direction as the speed of the adventurer will be mentioned already. By flipping the sprite, the adventurer can now face the direction he is moving. Fig.3. shows that when the player is trying to move, he plays his run animation.

c) Adventurer's Jump: Like the movement, the animation for the adventurer jump is created. An empty object is created and fixed on the bottom of the adventurer to check if the place they land is a ground. This allows the player not to stuck in the edge of a platform and make sure they fall. The jump is made that the longer the space key pressed the higher the adventurer jumps. Fig.4. displays the jump mechanism when it is used.

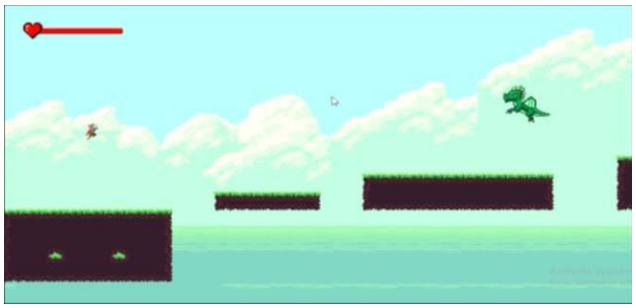

Fig. 4. Disappear Game Player's Jump Mechanism

d) Adventurer's Attack: For attack, the adventurer has a wide variety of options. The adventurer has three sword attack and a jump slash attack animation. The three sword attack needs to performed in sequence combo if the player chooses to stop the attack in the middle it will reset the combo count. When hitting more than one enemy the adventurer hits all of them. The air attack slash has a cool down which makes it difficult for the player to spam it. When the player tries to air attack immediately after the first air attack the cool down make it impossible to transit into the air attack state again. The cool down makes the player to wait a certain amount of time before using the air attack again. Fig.5. presents one of the Player's combo attacks.

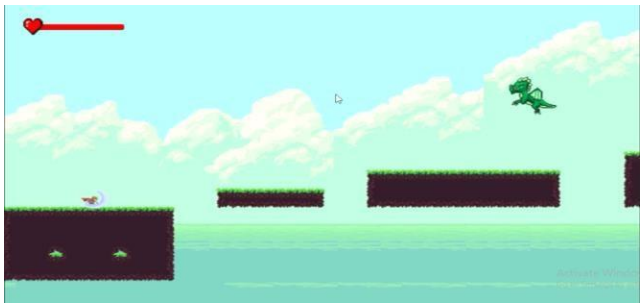

Fig. 5. Disappear Game Player's Sword Attack

e) Adventurer's Archery: The adventurer can shoot the enemy using the bow and arrows, where arrows are limited. When the arrow hits enemy its deals damage if not it disappears in the air. Player can jump and shoot their arrow. The arrow will be shot on the direction that the adventurer facing and shoots on a straight line. The arrow is not part of the adventurer it's another game object created during the ranged attack. When the adventurer shoots the bow the arrow spawns there and goes in the direction it got shot at. Fig.6. displays the archery of the player while he is jumping.

Fig. 6. Disappear Game Player's Archery

f) Adventure's Health: The health of the adventurer is shown in the corner of the screen in a bar. The adventurer will get hit when he was attacked by the enemies. When all the health is reduced to zero, the whole level will be reset. Throughout the game there'll be also health portions left which will increase health of the adventurer. Also the fire ball from enemies can hurt the player. When the adventurer accidently falls out of the platform and out of camera's view it's taken as the adventurer lost their whole health. When the adventurer losses all of their health the whole stage that they were in and will get restart again. All the enemies that died before will be reappearing again.

E. Manager's Code

Game also needs managers that will help the Game to operate properly.

a) Game Play Manager: Disappear has the game flow separated as stages. When going from one stage to another there needs to be some conditions fulfilled like defeating all the enemies or reaching the end of the stage. When that happens the game play manager will change the stage to the next stage.

b) Sound Manager: In each stage there needs to be sound effects. The sound manager makes sure to give the game a lively vibration. In each stage the sound game object will be carried to the next stage. It also gives the game background music by taking a set of sound records also the sound of the adventurer's footsteps, sword slashing sound etc.

c) Camera Shake Manager: Cinemachine virtual camera is used to make the job of the main camera easy.

Fig. 7. When Player gets hit, the screen turns red and white to create an impact of getting hurt

Using this we can make the main camera follow the player in a particular distance. When the player gets hit the screen will go to red for a second and shakes which gives the idea of adventurer getting hurt to the player. At that time the

enemies can't attack him neither the adventurer can attack the enemies. There are animations to play when the adventurer gets hit and stuns for a second that causes the camera to go white and red and another animation which shakes the whole camera to create impact on the player is shown in the Fig.7.

F. Skeleton's AI Design:

a) Skeleton Creation: Like adventurer the sprites are created using Aseprite or got from free pixel creation. The Skeleton also gets attached with the Rigidbody2D and Box Collider in the Unity. The pixel size of all the skeleton sprite has been changed to make the skeleton look bigger than the adventurer. The sprites are cut into multiple and sliced using Sprite Editor then dragged into the animation tab to create animation. The total of sprite sheet present is patrol, attack, hit, detect and death for each skeleton separately.

b) Skeleton's Movement: Awake function in the skeleton refers the Rigidbody2D and also gets the player's position. FixedUpdate function triggers the skeleton to start patrolling the area the moment the game started. Like the adventurer when it turns from left to right the sprite is completely changed to face the opposite. The direction of the skeleton to move is controlled by changing a Boolean variable frequently. For the skeleton to patrol a particular distance from where it placed a simple Equation (1), (2) is used. The variables minX and maxX, the maximum and minimum the skeleton will be travelling, distance is the distance that skeleton needs to cover.

$$maxX = transform.position.x + (distance / 2) \quad (1)$$

$$minX = maxX - distance \quad (2)$$

Here transform.position.x gives the skeleton's position, that helps to calculate the maxX and minX. The division of 2 just shortens the distance the skeleton should travel. Fig.8. presents the skeleton movement while patrolling its area.

Fig. 8. Disappear Game Skeleton patrolling its area

c) Skeleton's Attack: While patrolling its area the skeleton will always check the game objects tag near it. When the adventurer comes near certain area of the skeleton it detects the adventurer's tag and stops for split second and starts charging towards him with a change in speed. If the adventurer got off guard by it may land a hit on him using Scythe or Morningstar depending on the skeleton that wields them which causes the adventurer to stun for a moment and loss their health. Both Scythe Skeleton and the Morningstar Skeleton are melee attack enemies meaning they can only attack the player when they are in a closed range. But the Mage Skeleton is a ranged attack enemy meaning it can summon fire balls from very distant place it does not need to be close to the adventurer to attack them. Fig.9. shows the Morningstar Skeleton about to hit the player.

Fig. 9. Skeleton tries to attack the player using a weapon

d) Skeleton's Health: The Health of the skeleton varies for each type. The Scythe Skeleton has less health than the Morningstar Skeleton which has less health than the Mage Skeleton. When the adventurer hits the player the skeleton gets stunned. In this period of time the adventurer's extra hits won't cause damage to the skeleton to prevent the player from scamming sword combos. When the health of a skeleton reduced to zero the skeleton will stop its functions and falls on the ground crashing which is caused by isKinemic function check that is shown in the Fig.10.

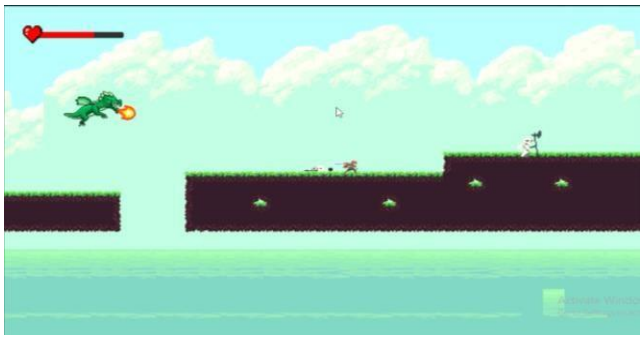

Fig. 10. Player kills the skeleton making it to die

G. Dragon's AI Code

a) Dragon Creation: Dragon's sprite sheets are created using a pixel art editor with green colour palette. The animation is simple by only changing small details in the sprite. When it's added in the Unity engine additionally the Rigidbody2D and Collider components are added to its game object. The sprite sheet that is used to create the dragon is idle flying, hit, attack, fireball and death.

b) Dragon's Movement: Like in skeleton Awake function refers to Rigidbody2D, Animator and adventurer's position. When the dragon is in its idle state it flies in its position in vertical movement. When the dragon reaches a certain height it flies up and down relative to its position in Fig.11.

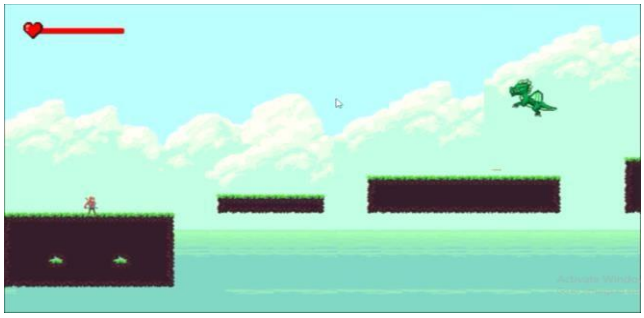

Fig. 11. Disappear Grame Dragon flying in a station

c) Dragon's Attack: If the dragon detects the adventurer in a particular radius it stops in one position and starts spiting fire balls at the direction of the player. When the Attack function gets triggered and creates a fire ball near its mouth, the dragon's attack has its cool down to make sure the fire ball to create rapidly. This gives the player to change their position in time. If the player stays in the position for long the fire ball might hit him. This is presented in the Fig.12. in which the dragon is about to hit the player using its fireball.

Fig. 12. Dragon tries to attack the player with its fireball

d) Dragon's Fire Ball: The fire ball itself is another game object the arrow for the adventurer. The ShootThePlayer function is called when the dragon detects the adventurer. The Fire ball gets the position of the adventurer in the XY axis and heads toward him to cause damage. If the fire ball did not hit the adventurer or took long time it disappears in the thin air. Fig.13. displays the fireball coming at player's direction.

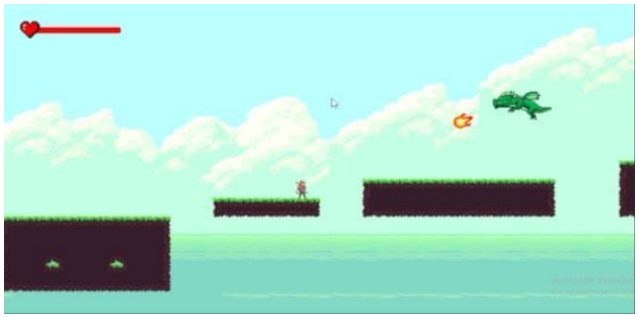

Fig. 13. Dragon fireball directly goes for the player

e) Dragon's Health: Like for the skeleton when the adventurer hits the player the dragon gets stunned while flying. When the health of a dragon's health reduced to zero the dragon will stop its functions and falls on the ground crashing which is caused by isKinemic function check presented in Fig.14.

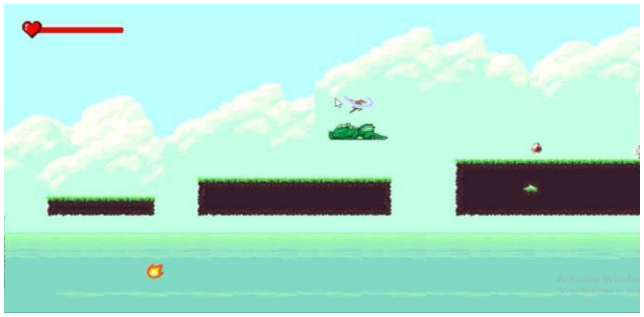

Fig. 14. Dragon was killed by the player

H. Disappear Game with FSM AI Model:

The FSM AI in this game has given a simple yet logical solution to the Enemy AI. Enemies patrolling the game space are made to keep their patrol, until they notice Adventurer has entered in their territory and start going after him. When the Enemy goes near the Adventurer, it starts attacking them until they kill the adventurer or get killed by them. If they lost the adventurer, they will cut the chase and start patrolling on the area where they lost the Adventurer. The Enemy AI therefore makes it hard for the Adventurer to go any further without facing it. This change in strategy will be great advantage for the enemies or the Adventurer depending on the position it changed its location. Depending on the pre-defined situation the enemy will be able to create decision whether to make a change in its routine or not.

I. Disappear Game Implementation in Unity:

This game is developed and implemented in the Unity Game Engine. Unity is the good choice for a game developer because it allows them not only to create 2D games but also 3D games as well. Both of the game mechanisms are mostly the same which makes it easier to work with. When Unity is the first to opens, it gives an environment where all the game object can be dragged and dropped. All the process working except for the program can be finished here. In Unity, all the game object's attributes like position, size and extra added component can be added by simply mentioning it or changing the value in textbox. Unity has a different section to show the working of things that will be able to seen by the player other than the things seen by the game developer. It helps to test the game in a given interval of improvement. It also shows if there is any error occurred during the game loaded. Unity helps in the process of making the sprite into an animation with its time frame. It makes the created animation into a flow of order which helps when the state needs to be used. It also helps in the process of making painting of the grid with appropriate tiles. Unity now allows deploying to 27 distinct target platforms. Android, PC, and iOS are the most popular platforms. Unity's free license makes it accessible to game creators all over the world.

V. CONCLUSION

This paper explains the creation of video game development with a simple FSM AI using Unity Game Engine. With ever growing game industry that has lots of entertainment produced it's always rare for a meaningful message game to appear ones in a blue moon. Like any FPS (First Person Shooting Game) there is a big audience for story based games and simple 2D games. As a 2D game it has a possibility of any person due to its simplicity nature. The enemies created using "Finite State Machine AI" are pre-programmed that they can face the predicted situation. This game is a story driven adventurous action game that has enemy's that can make decision upon the change of a situation. But even if the player and enemy functions of the game has been implemented, this is still an incomplete work in the eyes of a gamer. Thus a 2D game is created and developed that will bring awareness about mental health issues.

The future focus in the work could be, new environment tiles and contrasting color palette can be used to show the adventurers emotions. Backgrounds and a reacting world object will make the player to explore the world more. New NPCs can be added that can be not a threat to the player. The game needs to add more sound to make exploration more alive. Using Unity's particle system, we can also add snow, rain and fog in the back ground which will make the players travel through the game challenging. There can be platforms that moves or disappear which will make the player to more careful on the game itself. FMS that is applied here could also be used in the real world situation where one needs to create realistic simulations of software and cyber security.

REFERENCES

[1] Pichit Promsutipong and Vishnu Kotrajaras, "Enemy Evaluation AI for 2D Action-Platform Game", 14th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2017

[2] Apostolos Melinones and Ioannis Plas, "Developing Video Games with Elementary Adaptive Artificial Intelligence in Unity", Intelligent Systems Conference, 2017.

[3] Antonio Sag and Tihomir Orehovacki, "Development of 2D Game with Construct 2", 42nd International convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2019

[4] Xuesen Lin, Member of IEEE, "Multi-Behaviors Finite State Machine", IEEE Youth Conference on Information, Computing and Telecommunication, 2009

[5] Ivan Bravi, Diego Perez-Liebana, Simon M. Lucas and Jialin Liu, "Shallow Decision-Making Analysis in General Video Game Playing", IEEE Conference on Computational Intelligence and Games (CIG), 2018.

[6] Sule Yildirim Yayilgan and Sindre Berg Stene, "A survey on the need and use of AI in Game AI", Proceedings of the 2008 Spring Simulation Multiconference, SpringSim, 2008.

[7] Mattias Edlund, "Artificial Intelligence in Games: Faking Human Behavior", Computer Sciences Human Computer Interaction, Independent thesis Basic level (Student thesis), 2015.

[8] Jie Hu, Wang gen Wan and Xiaoqing Yu, "A Pathfinding Algorithm in Real-time Strategy Game based on Unity 3D", International Conference on Audio, Language and Image Processing, 2012.

[9] Ying Fung Yiu and Rabi Mahapatra, Computer Science and Engineering Department, USA "Regions Discovery Algorithm for Pathfinding in Grid Based Maps", Second International Conference on Tansdisciplinary AI, 2020.

[10] Umar Affandi Shahrin Iskandar, Norizan Mat Diah and Marina Ismail, Malaysia, "Identifying Artificial Intelligence Pathfinding Algorithms for Platformer Games", IEEE International Conference on Automatic Control and Intelligent System (12CACIS), 2020.

[11] Geoff Skinner and Toby Walmsley, "Arificial Intelligence and Deep Learning in Video Games A Brief Review, IEEE 4th International Conference on Computer and Communication Systems, 2019.

[12] Qijin Sun and Suoju He, "Artificial Neural Network Using the training set of DTS for Pacman game", 11th International Computer Conference on Wavelet Actiev Media Technology and Information Processing(ICCWAMTIP), 2014.

[13] Rebecca Brown and Curry Guinn, "Developing Game-Playing Agents That Adapt to User Strategies: A Case Study", IEEE Symposium on Intelligent Agents (IA), 2014.

[14] Aswin Thomas Abraham and Kevin McGee, "AI for Dynamic Team-mate Adaptation in Games", IEEE Conference on Computational Intelligence and Games(CIG'10), 2010.

[15] Paul A. Rummell, Canada, "Adaptive AI to Play Tower Defense Game", 16th International Conference on Computer Games (CGAMES), 2011.

[16] Lenonardo Jeanny Praganth and Darius Andana Haris, "Serenade Tower Hack and Slash Game", IOP Conference Series: Materials Scienece and Engineering, 2020.

[17] Reza Andrea, Sefty Wijayanti and Nursobah, "Finite State Machine Model in Jungle Adventure Game an Introduction to Survival Skills", I.J. Information Engineering and Electronic Business, 2021, 4, 55-61.