A NOVEL APPROACH ON NEXT GENERATION GAMING

Tarun J[1], Harsha Sai Kumar V[2], Anwar Basha H[3], Vidhyasagar B S[4]
[12]UG Scholar, [34]Assistant Professor,
Department of Computer Science & Engineering,
SRM Institute of Science & Technology (Deemed to be University),
Vadapalani, Chennai, Tamilnadu, India.

ABSTRACT: A game is an electronic sports that involves interaction between the user and bots with advance technology in artificial intelligence, Virtual reality and Augmented reality. This game is based on first person shooter(FPS) , An Iron-Sights view. The main focus of this project is to make an AI bot in games to provide a challenging gameplay to the hardcore players. It's a "COGNITIVE GAMING" procedure where player's mind is tested with challenging and adaptive AI bots. The "Unity" engine which provides real time interaction with players. The proposed system allows a new terrain(with VR), new characters and new weapons. For the shortest path traversal A* algorithm is used which does a perfect guesses between waypoints which in turn forms a waypoint network.

Keywords- Game development , Artificial Intelligence, Cognitive Gaming, logic programming and scripting, Virtual reality, Augmented reality and FPS.

1. PATH TRAVERSING USING A* ALGORITHM

The **Path Traversing** of the NPC's (non player controller), the AI bots, is done using the C# script code and in-built module of the game engine called "Unity". It uses the AI Algorithm called the A*, which is best suited for guessing the best path for traversing, when compared with the other path traversing algorithms. In this algorithm navigation graph is to get 'f 'score, 'g' score and add them to find the shortest distance to the target in the navigational mesh called Navigational paths(a list of waypoints).

The path is searched by the agent using the A* algorithm in the navigation graph, this is the modern path finding systems which at least contains four parts.

Navigation Path: Usually complied at development time. Describes all the traversable areas and their connectivity.

Agent Path Queries: Performs searches on the Navigation Graph to find paths (a list of waypoints) to steer towards in sequence to get the agent to its destination.

Agent Path Steering: Updates the Agent's position over time to move it through the list of waypoints (the path) returned from the Path Query step.

Local Avoidance: As other dynamic objects (such as other agents) will not represented in the Navigation Graph, a separate system must exist that moderates the queried path to perform on-the-fly adjustments to the agent's velocity to avoid nearby dynamic objects.

By using these four parts in the process of finding the path, the Agent traverse in the path by find the nearest waypoints by avoiding the obstacles in the Navigation Path.

A* Search Algorithm:

It is complete. If a solution exists it will be found. An heuristic is used to determine which neighbor to search next. This significantly speeds up the search by trying to visit as few nodes as possible. It does this by scoring neighbors as they are encountered based on their likeliness to lay on the cheapest path. Assuming an admissible heuristic, will always find the cheapest path.

A* Pathfinding:

The Navigation Mesh have grid squares. The center of the square is the node. The Diagonal shortcut Method is used to calculate the lowest distance. A* is what it is, what we do to the neighbor nodes. As we add them to the open list and calculate a score for these referred as ‘f’ score which is basically a guess how likely this node brings us to the target node on the optimal path. The neighbor node with lowest ‘f’ score is considered as best candidate in the next iteration of search loop. This would generate cheapest path.

Open list: Contains nodes we have discovered as neighbors during the search but have not yet processed. We select one node to process from the open list with each iteration of the search loop and remove it from the open list once processed.

Closed list: As we select nodes from the open list, we remove the open list and add them to the closed list. This allows us to keep track of all the nodes we have already processed so we don’t try to process them again.

Node Cost Function: This function is used to calculate the ‘f’ score by the addition of the ‘g’ score and the ‘h’ score. It will find the cheapest path using iteration of the search loops.

$f(n)=g(n)+h(n)$

g(n)=real cost to current node from initial node.

h(n)=heuristic Estimate to goal from current node.

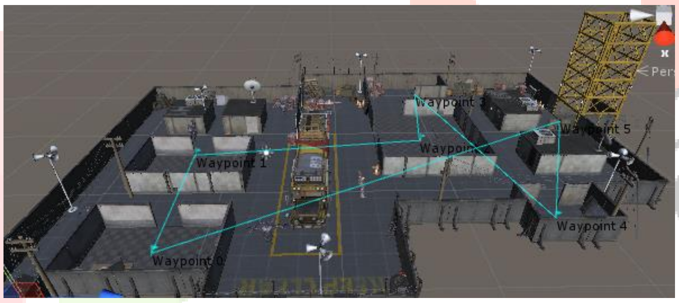


Figure 1.1 Connection of waypoints – A Waypoint Network.

2. CHARACTER DESIGN

For the character design we do our research how the character must be used and what the character must do (the Non player controller). This Non player controller is an AI bot which will not be controlled by the player, these bots will have Artificial intelligence’s adaptive learning with dynamic coding. These bots have been given instruction by coding the activities that has to be done like moving around the terrain, triggering, attacking, etc., the bot intelligence is tested in a way that it traversing with humanly behavior to the waypoints(path) given in the environment. The bot’s intelligence is literally equal to the intelligence of the player so that the player can feel the game challenging. As the difficulty level of the bot increases the hand-mind coordination and concentration of the player increases.

The First Person Shooter (FPS) is the main character in this game which will have the cognitive abilities, this character is completely controlled by the player, the player can see the terrain from the eyes of the FPS. The FPS is having the iron sight view to zoom in the target. This is a 3D character which the player make it to do humanly actions like jump, crouch, fire, etc., this helps the player feel the game realistic. The game play is completely in the hands of the player. The FPS can communicate with the objects in the game like Audio tape's, TV's, etc.

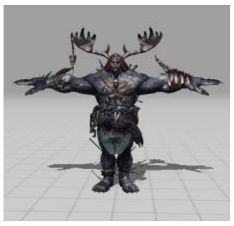


Figure 2.1 Chaos Devourer

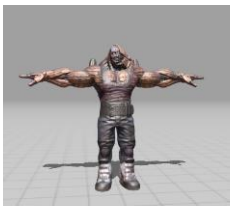


Figure 2.2 Phantom Knight

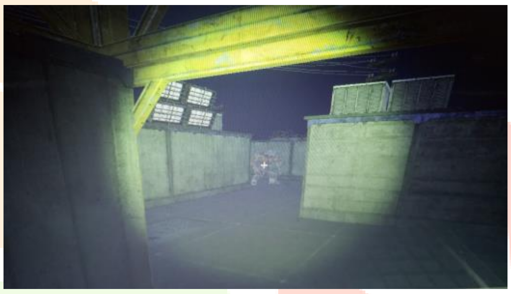


Figure 2.3 First Person Shooter with Flash

3. TERRAIN DESIGN

To design the terrain first we need to take reference that how the terrain looks like, what are the objects to be added and the paths to be traveled, we need from a prototype to the terrain then we develop the original terrain. The terrain is the environment where the player and the bot traverse from one place to other. The terrain is the layout having waypoints and the solid objects which are static, these objects can't move with respect any of the sub systems and player, bots could not pass through these objects. This static is an inbuilt option in the unity, these another option called the mesh collider which will from the wall around the objects so that the player, bot's can't pass through them.

The terrain is having the walls, doors and objects which are static. The navigational mesh obstacle is use when a new object is created so that the object is removed from the navigation path.

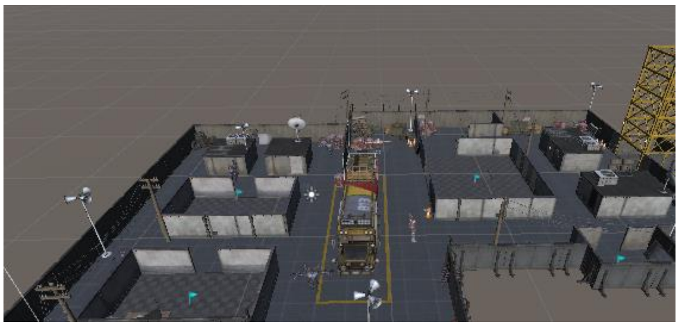


Figure 3.1 The Terrain

3.1 NAVIGATIONAL BAKING SYSTEM

The Navigational Baking system is the process where it can be used for real time lighting effects and baking a mesh of any solid objects. Instead of changing lighting and intensities , it's better to set lighting and bake it so that it stays constant in the entire scene when the light source is deleted. This baking system also has another important role which is providing NPC's not to collide with solid objects.

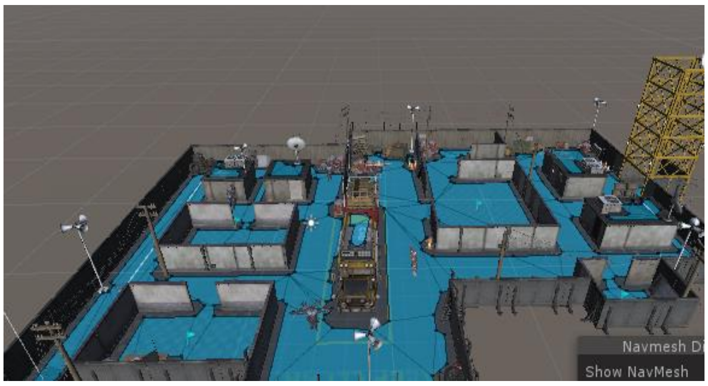


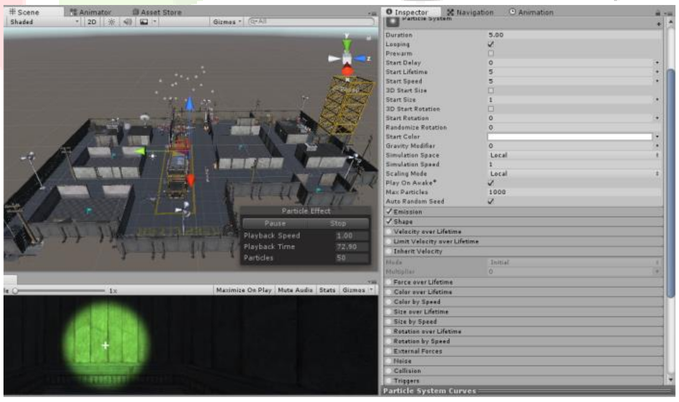
Figure 3.1.1 The Terrain with Navigational Mesh

We have height mesh in unity. Whenever we change the step height, the height mesh changes. This is used to climb the objects of certain height which are less than the step height. Whenever the player try to climb the object this height mesh will allow the player to climb object without passing through them.

This navigational mesh decides through which objects the players should pass and through which the player shouldn't pass. This will create the path which the player can traverse. This is the important concept in the game development, this will help in avoiding collision with the objects in the game by avoiding the navigation static objects in the terrain.

4. PARTICLE SYSTEM

In a 3D game, most characters, props, and scenery elements are represented as meshes, while a 2D game uses sprites for these purposes. Meshes and Sprites are ideal way to depict "Solid" objects. There are other entities in games however that are fluid and intangible in nature and consequently difficult to portray using meshes or sprites. For effects like Blood Dispersions, Tsunami, Smoggy clouds, flames, magic spells, rain, Fireworks and other effects is done using the particle system in the game engine.



Duration	5.00
Looping	✓
Prewarm	☐
Start Delay	0
Start Lifetime	5
Start Speed	5
3D Start Size	☐
Start Size	1
3D Start Rotation	☐
Start Rotation	0
Randomize Rotation	0
Start Color	
Gravity Modifier	0
Simulation Space	Local
Simulation Speed	1
Scaling Mode	Local
Play On Awake*	✓
Max Particles	1000
Auto Random Seed	✓
✓ Emission	
✓ Shape	
Velocity over Lifetime	
Limit Velocity over Lifetime	
Inherit Velocity	
Mode	Initial
Multiplier	0
Force over Lifetime	
Color over Lifetime	
Color by Speed	
Size over Lifetime	
Size by Speed	
Rotation over Lifetime	
Rotation by Speed	
External Forces	
Noise	
Collision	
Triggers	

Particle System Curves

Figure 4.1 Particle System

4.1 DYNAMICS OF THE SYSTEM

Every particle has lifetime during which it can undergo various changes. It begins its life when it is generated or emitted by the particle system. The system emits particles are random positions with various shapes and angles and also arbitrary meshes .When the time is over particles disappears immediately or within few seconds after the expired time as it posses a effect of slow disappearance i.e. Fade out or Fade in.

The choice of emission and average particle lifetime determines the total number of particles in the stable state.

4.2 DYNAMICS OF THE PARTICLE

The emission and lifetime settings affect the overall behavior of the system but still the individual particles can also change over time. Each particle has **Velocity** vector to determine its direction and the distance of particles moving with

each frame update. These velocities can be changed by forces and gravity applied using the particle system or when the particles are blown around by the wind zone on a terrain. The color, size and rotations also changes over lifetime. Here particle uses a red color for providing blood dispersions when the player or the NPC bots are hurt. These are being trigged according to the number of hits given by the player or the bots.

5. THE GAME ARCHITECTURE

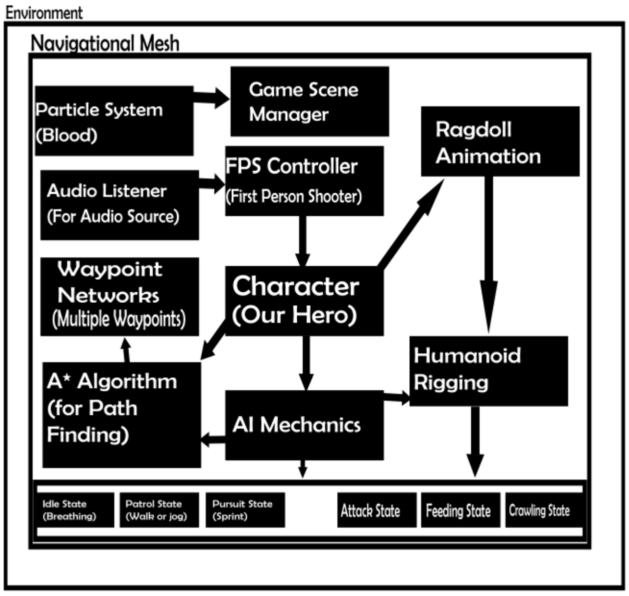


Figure 5.1 The Overall Architecture

5.1 GAME SCENE MANAGER

A Game scene is a state that contains environment, obstacles, decorations etc., In each Game scene we place our environments, obstacles, etc., for essentially building and designing the scene. The player can do teleport from one scene to another scene. To manage everything in that we need a manager, it is game scene manger. The game scene manger manages how the particle system should behave on various scenes eg: in one scene it will be day and in the other scene it will be night, in other it rains. Taking control over multiple scenes and the records(scores) and achievements. The game scene manager is having static properties and static methods to manage the scenes like the loading scene in between the game or before a new mission or after the mission ends. i.e. the scene which can’t be controlled by the player except for skip scenes.

5.2 PHYSICS MECHANICS

Physics plays an important role everywhere, not only in reality, even in virtual world physics can never be ignored or destroyed. Here physics mechanics like gravity, Sound waves, triggers, force, velocity etc. have been added for the purpose of making some realism in a virtual gaming world. Blood splashes when any of the character is hurt, here blood doesn’t fly in the air, with the gravitational force of 9.8 m/s it falls down.

5.3 AI MECHANICS

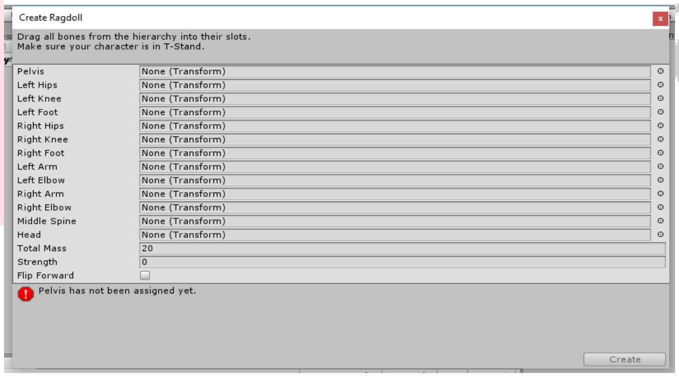
Artificial Intelligence, the most important thing to be considered for making the character to actually work like a human. Providing intelligence to machines and make it to adapt human characters is tedious process. Here the NPC bots when they see a human , they get triggered and runs toward him to perform the action of attack. Even when an enemy is behind , with the sound of the boots, those NPC's immediately get triggered to make a about turn. When a sound of a gun shot is heard by the NPC's within the radius, it gets triggered and run towards the sound source for investigation the area. Even during the flashlight on the NPC's the AI Mechanics takes place.

5.3.1 AI MECHANICS STATES

Attacking : The Bots perform an attack like punching, swiping, combo attacks, Biting.
Crawling : The bots perform a crawl when they are at an hurt threshold.
Sprint, Run, Walk : From fast movements to slow movements.
Breathing : The Idle state.
Feeding : The NPC bots eats dead bodies when satisfaction rate is below 100.
Thinking : The bots thinks for a moment before getting triggered by something and investigating the area.

5.4 RAGDOLL ANIMATION

Animation provided with the concept of
Humanoid riggings or otherwise called puppet warps. These animation provide some realistic action like a human where all the joints in our body are used to bend individual parts. i.e hand could be acted like, only the wrist can move keeping other parts fixed, then elbow part, keeping shoulder joint fixed and so on. That's the reason it is named as a ragdoll where a puppet doll has joints, which anyone can make individual part act like a human and play with it. Here when the NPC's are shot in the knee, it just doesn't fall down, it bends the knee then falls down.



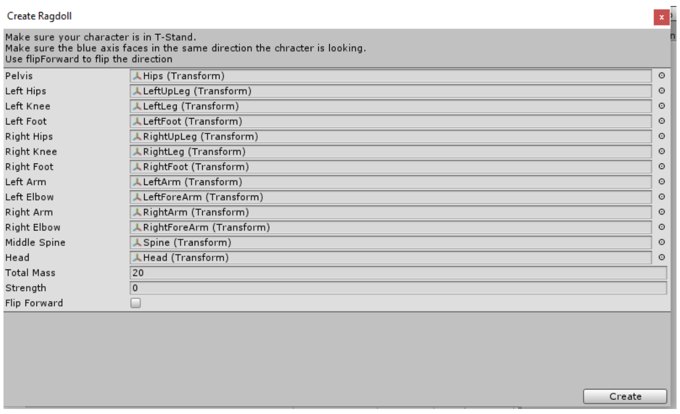
Create Ragdoll

Drag all bones from the hierarchy into their slots.
Make sure your character is in T-Stand.

Pelvis	None (Transform)
Left Hips	None (Transform)
Left Knee	None (Transform)
Left Foot	None (Transform)
Right Hips	None (Transform)
Right Knee	None (Transform)
Right Foot	None (Transform)
Left Arm	None (Transform)
Left Elbow	None (Transform)
Right Arm	None (Transform)
Right Elbow	None (Transform)
Middle Spine	None (Transform)
Head	None (Transform)
Total Mass	20
Strength	0
Flip Forward	☐

Pelvis has not been assigned yet.

Figure 5.4.1 Before adding Ragdoll Setup



Create Ragdoll

Make sure your character is in T-Stand.
Make sure the blue axis faces in the same direction the chracter is looking.
Use flipForward to flip the direction

Pelvis	Hips (Transform)
Left Hips	LeftUpLeg (Transform)
Left Knee	LeftLeg (Transform)
Left Foot	LeftFoot (Transform)
Right Hips	RightUpLeg (Transform)
Right Knee	RightLeg (Transform)
Right Foot	RightFoot (Transform)
Left Arm	LeftArm (Transform)
Left Elbow	LeftForeArm (Transform)
Right Arm	RightArm (Transform)
Right Elbow	RightForeArm (Transform)
Middle Spine	Spine (Transform)
Head	Head (Transform)
Total Mass	20
Strength	0
Flip Forward	☐

Figure 5.4.2 After adding Ragdoll Setup

6. CONCLUSION

Here we are, into one of most addictive technology in future, it's the Virtual reality. So the idea of the project is bringing 3D FPS games in VR for addictive hardcore gamers out there. This could be really challenging and fun for the e-sports players where it's totally new playing without the most important gadgets used till now, The mouse and Keyboard.

7. REFERENCES

1. *Cognitive Gaming* : Wei cai, Yuanfang Chi and Victor C.M. Leung, The university of British Columbia.2017

2. *Researching on AI Path-Finding Algorithm in the Game Development* : Xiangguange He, Yaya Wang, Yanyan cao.2012

3. *Research on Intelligent 3D Path Finding in Game Development* : Miao Wang, Hanyu Lu – Chengdu University of Technology, China.2012

4. *Learning VR Game Development Towards Software Basic Profile* : Waraporn Jirapanthong, Karuna Yampray, Datchakorn Tancharoen, Dhuraki Pundit University, College of Creative Design and Entertainment Technology.2017

5. *Practical Game Design and Development Pedagogy* – Paul J Diefenbach : Drexel University.2011

6. *Introductory Game Development : A mix of programming and art :* Chao Peng – University of Alabama in Huntsville,USA.2015

7. *Analysis of Game Development Activity using Team-Based Learning :* Akiko Teranishi , Minoru Nakayama Wyeld and Eid A Mohamad- Tokyo Institute of technology, japan.2017