# Portfolio website

First Name: Adeogo

Last Name: Adeniji

Age: 21

## Introduction

I'm a graduate student from Staffordshire University, specializing in computer games development. I have a strong passion for creating immersive gaming experiences. I have experience programming in languages like C++, C#, and Java, as well as using Unity and Unreal Engine. Additionally, I have a solid understanding of computer science principles, including graph theory, network flow algorithms, dynamic programming, and sorting algorithms. This knowledge equips me to tackle complex problems and find innovative solutions.

I also specialize in AI development, with knowledge in areas such as steering behaviors, pathfinding, decision making (finite state machines, behavior trees, G.O.A.P, fuzzy state machines), and more. I am continuously expanding my knowledge and aiming to further enhance my expertise by reading books and practising. Alongside AI, I possess knowledge in physics and mathematics, actively working on improving my skills by developing my own physics engine in opengl and SDL. Concepts like Newton’s Laws of Motion, vectors, collision detection/response, and car physics are among my areas of proficiency.

I am always eager for new challenges and opportunities, I am confident in my ability to deliver high-quality work and contribute value to any team within the dynamic and ever-evolving gaming industry.

## Skills

C++, C#, Java, Unity, Unreal

### Unity

As someone who has been using Unity for 3-4 years, I have a strong understanding of the platform and are familiar with many design patterns. My proficiency in Unity enables me to create high-quality games with engaging gameplay and visually appealing graphics.

I am able to utilize the various features and tools of the engine to bring my game ideas to life and have the ability to troubleshoot and solve problems as they arise. My extensive experience with Unity makes me a valuable asset to any team in the gaming industry, and I am well-equipped to take on complex projects and deliver exceptional results.

### Unreal

With my strong foundation in programming and game development, I have quickly gained proficiency in Unreal Engine, using its various features and tools to create high-quality games with cutting-edge graphics and immersive gameplay. My ability to adapt to new technology and learn new platforms quickly has allowed me to become proficient in Unreal Engine in a short amount of time, and my understanding of the engine's code structure enables me to troubleshoot and solve problems efficiently.

### SDL and Opengl

My skills and ability using OpenGL and the SDL library are impressive, as I have a strong understanding of both technologies and are able to use them to create games. My proficiency with OpenGL and SDL enables me to create games with quality graphics and smooth gameplay, and I am able to utilize the various features and tools of these libraries to bring my game ideas to life.

Furthermore, I have hands-on experience in developing AI projects, game engines, and physics projects using OpenGL and SDL. This background has further strengthened my skills and practical knowledge in these areas

### General purpose

As a general purpose programmer, I have a strong foundation in coding and problem-solving skills. My extensive knowledge in graph theory has allowed me to solve many complicated problems, and my proficiency in network flow algorithms has helped me tackle a wide range of challenges. Network flow algorithms have been used to optimize transportation and distribution networks, schedule production in factories, etc.

### Maths and physics

I bring valuable math skills to game programming. With a deep understanding of complex mathematical concepts, I can create immersive gameplay experiences and find innovative solutions. My proficiency in math and ability to think critically and creatively allow me to deliver high-quality work, whether it's a large-scale project or a focused task. Additionally, I have knowledge in physics and continuously enhance my expertise by working on my physics engine. I'm well-versed in fundamental concepts like Newton's Laws of Motion, coordinate systems, vectors, derivatives and integrals, mass, center of mass, moment of inertia, velocity and acceleration, collision detection and response, car physics, transformations, and matrices.

### Ai

My proficiency in designing and implementing advanced AI algorithms enables me to create intelligent and realistic non-player characters (NPCs) that significantly enhance the player's gaming experience. I specialize in a range of AI techniques, including steering behaviors, pathfinding, decision making with finite state machines and state machine design patterns, behavior trees, G.O.A.P, and fuzzy state machines. With a commitment to continuous learning, I am dedicated to expanding my knowledge in these areas.

### Gameplay Programming

I specialize in gameplay programming with a strong understanding of design patterns and clean code practices. I am well-versed in implementing design patterns such as the command pattern, object pooling, template pattern, observer pattern, state pattern, component pattern, and more. These practices ensure efficient and maintainable code for better game development.

Additionally, I possess knowledge in uniform sampling, event systems, and other relevant concepts. This enables me to create robust and responsive gameplay experiences. I am constantly expanding my skills and staying updated with the latest advancements in the field to deliver high-quality results in game development.

## Skills learning

Networking

Tools engineering for unity

## Projects

### Kill To Survive

**KILL TO SURVIVE**

This is a horror fps battle royale made in Unity.

#### Intro

This game is an FPS Horror Battle royale where no one wants you alive, not other players and especially not the wonderers/wanderers and titans. Your survival is vital but also a hindrance as there can be only one winner, one survivor. This is a Deadman’s Wonderland. But who wants you dead, you ask? No,no,no,no… It’s who needs you dead.

The wonderers/wanderers are beings that want you dead in order to eat you, but they are not stupid zombies they are very much smart, as they know how and when to run and whether or not to team up with others of their kind to kill you. The enemy of my enemy is my friend. They also decide on whether or not to follow Titans.

Then there’s you. I pity you. This is going to be a long day but that’s only if you decide you want to be a victim. The titans want to kill you, the wanderers/wonderers want to kill you and your own people want to kill you, which is scary, but you can defend yourself and survive but not without massacre and bloodshed. You are given a pistol to survive and with that pistol comes infinite ammo and hope. You use that pistol to kill people and creatures in order to get money because you can’t pick up other people’s weapon. So, to get new and better weapon you buy it with money and to get money, you kill.

The theme the game is built around is horror and survival and how you could get killed from anywhere at any time. So, your goal is basically to survive but to survive you need to be able to kill whatever is approaching you at any point and you never know when or where that could be, and in order to do this you need stronger and maybe even more weapons. The horror aspect is also fully shown by the footsteps, wonderers and titans aren’t stupid they are very light on their feet.

What also adds to the horror theme is that you also have to buy ammunition forcing you to ration you ammo and make decisions which is more important to you ammo or weapons. The survival aspect come from how you can only heal up to 60% of your health and to fully heal you need to buy healables, but the issue arises here. This is because you only have 6 slots, and the first slot is already occupied by your pistol. So, you have to choose between weapons, healables, ammo and throwables. This fully embodies the theme of survival for me because survival is all about make difficult decisions with rationing and scarcity, in order to help you in the long run.

#### Work

Singletons and Persistent Singleton - So I created a singleton template class that creates singletons from any class and the persistence singleton template class creates a singleton that is persistent through out all scenes. So the Singleton persistent class creates a template class that still inherits from monobehaviour. The T is for the class we want to pass in to make a singleton, you can also add in a type constraint where T is a component which is what i did that means we will only be passing in classes derived from components components is the base class for everything attached to a GO. Then I made a function within the singleton template class so that when other scripts call this singleton pattern they get that instance.

Helpers - I created helper classes that doesn't inherit from anything and what their purpose is, is to reduce repetition mistakes caused by entering strings because when you use strings and if the string is wrong you get errors but no error that explicitly tells you that it is because the string name is incorrect. So how the helper class does it is by creating a public const string that is equal to the correct string we are going to use and whenever you need it you call the class and the name of the string and I use this in many cases as this can be called anywhere at any time. it is also easier to remember what i need to pass. I also created many iterations of this for many use cases to make things easier to access and modify and also look neater.

Save System and Account - I created an account class that contains player details such as the users' account name, level, Xp, money and gems and etc the account also keeps player achievements which i kept to only a few. I then created a player manager that is a persistent singleton which will be used in every scene and because i haven't set up the facebook, google play and iOS SDK and it only handles guest. everything pertaining the current player is handle through this manager and this manager creates a new player account if one doesn't already exist and it also saves and load the player (assuming the player has already registered as a guest). I then create a save and load system that is much more reliable and secure than player prefs and what it does is it read and writes whatever is being saved into a json file using a specific path set by unity and when the player is loading the account details it create a new account and overwrites that account with the account that was read from the path given. However this is not fully secure as i found the exact file and it is modifiable. A solution to this would be to use a binary save system, a firebase save system or a NodeJS but for now json file is fine.

Weapon wedge wheel and wedge activator - For my game, I created a wedge wheel for the weapon system I did this because of its quick draw. So basically when the zone closes in and the game is coming to a close and enemies are closer there needs to be a smooth way to change between your weapons and quick too and I came up with the weapon wedge wheel and the way this work is you tap the semi-circle in the bottom middle of the screen and while your finger is down your whole inventory opens up and you can see the weapons, healables and throwables in your inventory, then you drag your finger to the item of your choice and when you release your finger on that item then becomes your active weapon and the image within the semi-circle changes to the active weapon’s image. This is done using event triggers and events. Event triggers are a unity component. So how this is done is when you click on the semi-circle the event trigger is activated and that event trigger is onPointerDown and then it invokes an event called activate wedge and what this does is tell each wedge to activate and then on each wedge there are 2 event triggers: onPointerEnter and onPointerExit and what this does is that when the player’s finger is on the wedge it increases the alpha and the exit changes the alpha then when you raise your finger the onPointerUp trigger invokes a method that deactivates the wedges and if the player’s finger was last on the wedge it sets that wedge to be the active weapon but if it wasn’t on a wedge when they lifted up their finger it just deactivates the wedge.

One of the key features of the game was its dynamic movement system, which allowed the player character to move in a fluid and responsive manner. This was achieved by implementing the state design pattern with Pushdown Automata and Hierarchical State. With this design pattern, the player's movement behavior seamlessly transitioned between different states based on the gameplay situation. For example, the character could smoothly switch between walking, running, crouching, and other movement states, providing a realistic and immersive experience.

To enhance the gameplay experience further, the game incorporated an event system that played a vital role in various functionalities. One notable aspect was the weapon switching mechanism. Players could seamlessly switch between different weapons they had acquired during gameplay. This feature not only added versatility to combat encounters but also allowed players to adapt their strategies based on the situation at hand.

Another important use of the event system was in the in-game shop. The shop was represented by a house that, when collided with, triggered the opening of a shop UI. Within the UI, players could browse different sections for purchasing items such as weapons, ammunition, and healing items. The shop utilized a user-friendly interface, enabling players to easily navigate through the available items and make their selections. Upon purchase, the selected item was added to the player's weapon inventory.

The system also supported the selling of items. If players decided to sell an item, the corresponding item would be deducted from their inventory, and they would receive a percentage of the item's cost as compensation. This feature added depth to the gameplay by allowing players to manage their inventory effectively and make strategic decisions based on their resources.

All item details, including weapon stats, ammo capacity, and healing properties, were stored in scriptable objects. This allowed for easy customization and tuning of the game's items without requiring complex code changes. The scriptable objects provided a convenient way to manage and access item information, making the game more scalable and adaptable for future updates.

### Rulers Route

### Swerve

### Around The World

### Leap Learn Live

### Grappler

### Chax Arcade

### AI in Unity

### AI in DirectX and OpenGl