**Games engine and scripting**

**By: Abdallah Salhab 21110181**

**Name and briefly describe three different game engines currently popular in the industry. How do they differ in terms of features and target audience.**

Unreal engine:

Unreal engine is a collection of developer tools for building interactive 3D games, simulations and visualizations developed by epic games. It is written in C++. It was originally made for first person shooter games but now it can be used for making AAA games, visual effect and movies, VR games, simulations and is used to create indie games. Its target audience vary from game developers to interior designers and architects to filmmakers and many others.

Features:

Visual scripting (blue prints): allows developer who does not know how to wright code thanks to visual scripting language called blue print that makes the objects interactive.

Cross platform support: can be deployed into a wide range of platforms such as; pc, consoles, mobiles and VR.

cinematic editing and animation tool called Sequencer: enables users to use a specialized multi-track editor to create in-game cinematics.

Godot:

It is a free open source game engine that is released under the MIT license, it is designed to create 2D and 3D games for the pc and mobile and the web platforms and it can be used for non games software like editors. Its a great place for beginners to start but can also be used by professionals.

Features:

Open source and free: completely free with a MIT license, encouraging for contributions and transparency.

Light wight and fast: made for performance and is good for resource constrained projects.

combines the scripting capabilities of GDScript, a language similar to Python, with the power of a powerful editor.

Unity:

Unity is a cross platform game engine for making 2D and 3D games, it is used to make games, interactive simulations and animations, the engine is written in C++ and allows the developers to write in C sharp, and it provides a power full graphical editor that allows to make a Senes without writing code. It can be used by teams, beginners, indie games developers and professionals.

Features:

Asset store: provides a thorough store for plugins, tools, and assets, allowing for quick development.

User friendly interface: Its user-friendly interface makes it easy to use even for inexperienced users.

2D and 3D support: capable of producing both 2D and 3D games, with specific tools and processes for each.

**Describe the early origins of game engines and their evolution from landmark titles in gaming history.**

At the biggening games engines where simple frame works,

Like in the game colossal cave adventure that was a text-based adventure game where the player explores caves to find treasures, it was the first game to be and example of interactive fiction in which players could type commands to control characters.

There also a game called space invaders that featured a system for game logic chat could be reused in other games, and this was an early hint of what games engine could be.

The rise of modular design(allows the reuse, combining and modifying of the game elements):

One of the first games to introduce this concept is called Rogue that used the concept of procedural generation, where the dungeons in the game are generated algorithmically, this what made replay ability possible and that is an important feature in games engines.

One of the early games to feature a virtual machine is Zork that was platform independent because it could run independently of the hardware.

The transition to modern games engines:

Ultima series is a game that evolved to have complex world building, NPC interactions and it pushed the development of reusable game codes and modular design.

ID software revolutionized game development by adding modifiability to games such as the DOOM engine that allowed user to make custom levels and mods.

It also featured advanced 3D graphics and multiplayer networking in the Quake engine, by introducing QuakeC scripting language it allowed developers to modify the games logic.

Commercial games engine:

Epic games created a highly capable games engine named Unreal engine that could support various game genres. Developers could now create complex gameplay mechanics thanks to the unreal script.

After that unity technologies introduced an accessible and versatile engine named Unity that was aimed at independent developers, it had a user friendly interface and an asset store and support for multiple platforms.

**Reflect on the future trends in game engine development and their potential implications for the gaming industry. How might emerging technologies such as cloud computing, procedural generation, and virtual reality shape the landscape of game engines and game design.**

Cloud gaming is a type of online gaming that enables games to run on a server and to stream the game output into the user’s device.

Cloud gaming enables games to handle more complex calculation and a larger game world, cloud gaming could offer infinite processing and storage capacity, more complicated visuals, larger multiplayer sittings and real time updates, this all because there is no hardware limitation. Also the games will be accessible to everyone in any device and games engines will need to optimize the streaming performance and latency.

Procedural generation is a method of creating data algorithmically and not manually, in games it can be used to create unique game worlds with minimal manual effort, and it can be used to make the game tailor for every gamer by making personalized experiences that adapts to the user input, games engines have to add advanced algorithms to generate the environments, quests, characters to make for the unique gameplay.

VR or virtual reality and AR or augmented reality will add a new dimension of immersive experience to the games it will also add a whole new way of interacting with the game using the hands and eye tracking, games engines will need to focus on optimizing the performance for the VR to insure high FPS and realistic physics.

**Evaluate the role of Artificial Intelligence (AI) and Machine Learning(ML) in modern game development. How are AI-driven systems enhancing gameplay experiences, character behaviors, and procedural content generation.**

AI and ML have streamlined the game development process itself; it improved the testing of the games by making it automated using AI algorithms that can rapidly test games, identify bugs and glitches faster and more efficiently. AI is also used in procedural content generation that can generate deferent paths, worlds, levels or items autonomously. ML algorithms can make characters learn from the players actions and adapt to their behaviors this makes for a unique gameplay for the player and it makes for a personalized experience that immerse the gamer more into the game

ML allows games developers to create better looking and higher quality games with less time and effort.

**Describe the features of Unity games engine and explain the purpose and operation of each feature.**

There are a lot of features in unity that helps the developer to develop a game easier and more professionally like:

Animation:

The animations system in unity allows developers to make deferent animations for the deferent characters and objects.

The animation feature and animator controlled in unity enable developers to create manage and change between deferent animations states, and the developers can use the animations window to create keyframes and blend the animations using state machine.

Asset management:

The asset management page enables the developer to control over who has access to the assets that are related to the organizations account.

Its purpose is to organize and manage textures, models, sounds and scripts and they all are organizing in the project window, these assets can be created by the developer or imported from deferent sources like unity asset stores, and they can be imported and loaded into the project.

Scripting:

It allows the developer to control and add interactions for deferent game objects.

Unity uses C# as the scripting language, each script has functions like start () to handle initialization, update () for frame updates and fixed Update () for physics updates.

The scripts in unity are created then attached to an object to change int behavior and interactions.

Cross platform deployment:

Unity games engine supports a wide range of platforms like consoles, windows, android and IOS... through the build setting window unity allows developers to configure and deploy their games in deferent platforms.

This enables the developer to spend more time creating the game rather than dealing with technical issues to have the game in deferent platforms.

VR/AR support:

This allows the unity developers to create games for VR and AR by supporting SDKs, using the XR interaction tools and components the developer can handle the VR and AR interfaces and their deferent interactions.

**Review Different published games with Unity and compare the gameplay mechanics and systems present in the reviewed games with the features available in the selected game engine.**

Monument valley:

It is an adventure puzzle game; the main character is a princess who wants to return the sacred geometry that she had stolen.

The game is a puzzle game and the environment changes to guide the player deferent optical illusions and impossible geometry.

Monument valley used a lot of unity’s features to create such an amazing game like:

Using 3D graphics and rendering to and isometric view and to create and amazing looking environments, the game also used scripting to make the environment interact with the player and to make the complex puzzles work and the player to move and interact with other objects, they also used animations on the player and the environment to make the game feel more lively, they also used the particle system to make the world and the movements in the world feel more immersive.

Kerbal space program:

This is a simulation game that enables the player to design and build rockets and to explore space with those rockets.

This game uses unity physics engine to make the simulation of rocket driving realistic, and they also used complex scripting to make the rocket controllers and the missions that the play must achieve and to handle complex physics calculations, they used the UI system to create the UI that enables the player to build the rocket, and to plan the missions and the flight operations, they also used the used the 3D graphics and rendering to create detailed rockets and environments.

**Part 2:**

**Evaluate the features you implemented in your prototype using the selected game engine.**

In my game I used a lot of the features that unity provides to make my game, they made the game work as it suppose to and much better, the features that I used are:

3D Graphics: my game is a 3D game that will make the playing experience more immersive and it gives the game a new dimension that the player can move and interact with.

Scripting: I used scripting to do a lot of the interactions and the movements in the game like, the player movements, the winning, losing pause mechanisms and the coin collection mechanism, I also used it to create an infinite road that the player can move on, and the deferent interactions with the enemies, objects in the game.

Animations: I used the animations to make the main character look more natural with the movements, so he jumps, runs and idles when not moving.

UI: I used the UI system to create the deferent pages (menu, pause page, game over page) and the deferent buttons that makes the game more controllable and more professional, and I also used it to show the player health and number of coins collected.

Particle system: I used the particle system to make some objects to be more realistic and more immersive like the fire in the fire pit.

Audio System: I used deferent audio elements that make the game more lively, more enjoyable, and to make sounds that are connected to deferent interactions in the game.

Asset management: I used the asset management to manage and organize the deferent assts I used in my game and to import assets from the asset store into my game to make the game have deferent elements, an enemy and the main character.

Community and support: I used the community knowledge to learn complex stuff to be able to implement them in my game and I used the forms and document that the deferent users publish to solve problems that I had in my game.

**Describe any challenges or obstacles you encountered during the prototyping process, and how you addressed them using the tools and resources provided by the game engine.**

In the game that I was building I wanted to have a health system that shows how much health the player has using UI and to be able to have a mechanism that makes the character loses, so my first thought was to make a UI that shows how much hearts left for the player; like Minecraft; so I firstly looked at the links provided to us to see if there was a link that talks about health system, but there was no links about that topic, so then I turned to the Unity community and the Unity tutorials and I started and I started with a video that was on YouTube and it was done in a bad way and it had a lot of mistakes so I deleted everything I have done to follow the video and started following a video of a famous developer that made a great in detail video about the health system and how to deal damage to the player, so I reflected that and changed a couple of thing into my game to make it fit my needs and it works amazingly.

Another instance where I used the unity tools to fix a problem that I had with the cannon ball shooting mechanisms in the game where a pipe shoots a cannonball, and the other pipe destroys the cannonball, and there was a problem with them where the pipes that destroys the balls did not do that and I had them already laid out on the deferent roads, and the problem with it is that the collider of the pipe was not in the right place and it would not destroy the ball, so I went to the prefabs and in the prefabs I change the position of the collider so all the cannons are fixed in one simple movement, and I also used the prefabs to fix a road that was instantiated without a collider that instantiates the next road, so in the prefabs I added a collider at the beginning of the road and saved it and It worked.

**Show examples of how you utilized the asset management tools in the game engine to import, organize, and integrate assets such as 3D models, textures, animations, and sound files into your prototypes.**

In my project I used the asset management tool in every step of making the game and it helped me to organize the packages that I imported from the unity asset store and my assets that I made like scripts, prefabs, music, textures and more.

I used the asset manager to keep all the imported assets from the unity asset store organized to easily navigate between the assets that I need, there are a lot of packages that I used to make my game like; mini chibi free demo which I used to make my main character and it had materials, models and prefabs. Another one is the RPG monster DUO PBR Polyart that I used the get the spiky slug and its animations from, and similar package that I used to get the cactus monster from and its animation. I also imported another package that is Te Kni Ko-free package that had prefabs for the coins that the player collect and their animations and many more power ups that developer can add to make the game better. And the Gridness studios package that had most of the farm props, the package contained prefabs, materials, meshes, demo scene.

Using the asset manager the organization was easy as I let every package content in a file and not scattered every ware and each file had a materials, prefabs, textures... files, and that made the process of going to take a specific prefab from a specific folder much easier.

And I also used it to organize my assets as I made a music folder that has all the music and sound effects and a scripts folder that has all the scripts I wrote and the materials that has all the materials I created and that prefabs folder that has the prefabs of the objects I made. And this made it easy to use the assets and to edit them and to organize all the assets.

**Part 3:**

**Describe the process you followed to create a development plan, utilize a GDD to stimulate the mechanics and gameplay from it.**

To make the development plan I had to think of the game that i want to make so I got the main idea and I had to think of the controllers that the player needs to play and the type of game; third person, first person... then the game and how it is going to be like and the deferent game mechanics and the deferent UI ‘s needed to make the game more proficiently and also the sounds of the game.

After that I started planning the project timeline and the deadlines of those projects and at the end I made a sketch to make my idea of the game clear.

**Demonstrate how you implemented key features and mechanics outlined in the GDD**

In my game the player moves using the controls that is in the GDD and the game play is the same as it was in the GDD as the player has to move away from obstacles and collect coins so he can win and I also added sound effects for jumping and taking damage and collecting coins as written in the GDD and I also added all the deferent UI pages that is needed like menu, pause page, game over page

And I tried to stay on deadlines to make sure I was with the GDD specifications.

**Part 4:**

**Explain your approach to testing the playable demo. What testing methodologies would you employ, and how would you ensure comprehensive coverage of gameplay mechanics, features, and user interactions. Provide examples of the types of feedback you would get from testers and stakeholders during playtesting sessions.**

To test my game, I wanted to make sure that everything is working correctly so during the development of the game and in every milestone, I would test the features that I made, I also used two deferent testing methods the first is unit testing that focuses to testing every feature or component alone and check if it works correctly and after making sure that the components work good alone I started with the System testing that is testing the complete game system as a whole and to make sure the game works as it spouse to in a complete environment, and I also tested the game with deferent potential users that gave me so feedback of the game there was some feedback about the main camera that follows the player, the camera has some delay to make the movements of the camera smooth but is was set too high that it felt that the camera was moving alone, and that I had to rise the camera to make the deferent obstacles easier to see and to make the player see the depth of the obstacle, another feedback was when I was watching the player play and when he fell of the road the game over page did not come up so I forgot to make a trigger that triggers the falling of the player in that road.

And I got feedback about the roads and how they looked similar to each other, so I added some unique features to some roads to make it more exciting to play, and in one of the tests the lava in the game did not deal damage to the player.

**Interpret peer-review feedback to identify opportunities not previously considered.**

Some of the feedback that I got from the users that I did not mention and would be good in the future or in the next version and I did non concsidered in this version are: to make a menu where the player can choose the character he wants to play in, and to make deferent levels with deferent themes, and to or to give the player deferent power ups like Mario to make the character jump higher or become stronger...

**Explain the process you followed to document and address bugs, issues, and feedback encountered during testing sessions. How did you prioritize and categorize feedback, and how were changes and updates communicated to you as developer?**

During the testing sessions I wrote and remembered all the feedback the users gave my and I took their feedback into account, and after the testing I would fix the bugs and problems that was encountered while playing, and to make sure it was fixed I would put myself in the same circumstances it happens in the first time to see if it was fixed. I categorized the feedback into three categories; things that made the game not function as it was supposed to, bugs that made the playing less enjoyable and feedback of features that can be added to the game later.

I did priorities the functional feedback more because to make sure the game plays as expected and did not have any bugs that would make the game crash, after that the playability and visuals feedback to make sure that game in actually enjoyable to play with after that the feedback of the future I can add to the game in the next version.

Any additional updated on the feedback of the game would be delivered to me by massages or by meeting with the user to inform me on the things that I can change in the game, and I would tell them of the things that I fixed on the game thanks to their feedback.

**Part 5:**

**Provide examples of how you would iterate on the playable demo based on feedback and testing results? How did you incorporate user suggestions, address usability concerns, and refine gameplay mechanics to enhance the overall experience?**

During my game development journey and with the deferent feedback from the testing that I have done on the game I went through deferent iterations to get me to the final result, so after the testing of the game and the receiving the deferent feedback, I would address all their concerns and I would add their applicable suggestion in to the game, and would do this until they are satisfied with the game.

To refine and enhance the game I would take their feedback, analyze what can be done to improve on that aspect and I would try my best to make it as they expect for the game to be.

The game has been through a couple of deferent iterations and the goal was to improve on the game using the users feedback and to add any of their suggestion and to refine the game for it to better for the users.

In my game I to make deferent iterations to my main character sometimes adding features to it and sometimes improving things that is wrong, so it code, and attributes and its position and prefabs had to be change in the deferent iterations to get to the final results, and that same thing happened to most of the features, objects and systems in the game.

**Describe any future enhancements or refinements you would consider implementing to further improve the game based on player feedback and iterative testing. Plan to prioritize and iterate on these improvements to enhance the overall quality of the game.**

The game is now amazing, but it can be better so in the future of the game there can be aspects to improve and things or features to be added like:

The game can be more enjoyable if the player can get power ups to make him stronger, bigger, faster and get more health, and I can accomplish that by adding deferent power ups that gives that player these abilities.

Although the roads in the game keeps instantiating randomly and there are deferent one’s, the game can be better with making the objects placement on the road randomized, so every road is the same.

The game needs a different feel every time the player advances so adding deferent levels with deferent color theme or props.

I can add the ability for the player to change the main character to a deferent one.

I can improve on the character movement to make it like the other games, so it is easy for first time players to play, not to say that the movements are bad because they are great.

My Future Plan:

Firstly, I can improve the player's movement.

After that add more levels to the game.

Next, to make the objects placement on the road randomized.

Then, to add deferent levels to the game.

After that, add deferent power ups the player can take.

Finally, I can add a menu where the player can change the character into a different one.