|  |
| --- |
| Experiment No.1 |
| Installation of Unity and Visual Studio, setting up Unity for VR development, understanding documentation of the same. |
| Date of Performance: |
| Date of Submission: |

**AIM** : Installation of Unity and Visual Studio, setting up Unity for VR development, understanding documentation of the same.

**OBJECTIVES:-** The objectives for setting up Unity and Visual Studio for VR development involve installing the necessary software, configuring VR hardware, scripting interactions, understanding documentation, testing and optimizing applications, designing VR-friendly interfaces, engaging with the community for learning, and ultimately publishing VR apps. These steps enable developers to create immersive virtual reality experiences while ensuring efficient development and problem-solving through documentation and community support

**THEORY** :

Setting up Unity and Visual Studio for VR development is a crucial step towards creating immersive virtual reality experiences. Here are the objectives you should aim to achieve during this process:

* Install Unity and Visual Studio:

Install the latest version of Unity Hub or Unity Editor.

Install Visual Studio with the necessary components for C# development.

* Create a Unity Project:

Set up a new Unity project or open an existing one for your VR development.

* Import VR SDKs:

Depending on your target VR platform (e.g., Oculus Rift, HTC Vive, or others), import the relevant VR SDKs or plugins into your Unity project.

* Configure VR Hardware:

Connect and set up your VR headset and controllers with your computer.

* Configure Unity for VR:

Adjust project settings to support VR:

Change the platform to the one compatible with your VR headset (e.g., Oculus, OpenVR).

Set up the player settings for VR, like stereo rendering mode, recommended resolution, and supported SDKs.

* Understand Documentation:

Read and understand the documentation provided by Unity for VR development. This includes:

Unity's official VR documentation.

Documentation specific to the VR SDK you are using (e.g., Oculus VR, SteamVR).

Documentation for any third-party assets or plugins you use for VR development.

* Basic VR Interaction:

Learn how to create simple VR interactions, such as grabbing and manipulating objects, teleportation, and basic user interface elements.

* Scripting for VR:

Familiarize yourself with C# scripting in Unity to implement custom interactions and behaviors in your VR application.

* Testing and Debugging:

Test your VR application on your headset to ensure it functions as expected.

Learn how to debug VR-specific issues, including using Visual Studio for debugging.

* Optimization:

Learn about VR performance optimization techniques to ensure your application runs smoothly and comfortably.

* User Interface (UI):

Understand how to create VR-friendly user interfaces and menus.

* Documentation Navigation:

Develop the skill to navigate through Unity and VR SDK documentation efficiently to solve specific problems or explore advanced features.

* Community and Forums:

Explore VR development forums and communities, such as Unity forums or subreddit communities, to seek help, share your progress, and learn from others.

* Continuous Learning:

Keep up-to-date with the latest advancements in VR development, as the technology is rapidly evolving.

* Publishing and Distribution:

Learn how to package and distribute your VR application on the platform of your choice (e.g., Oculus Store, SteamVR, or others).

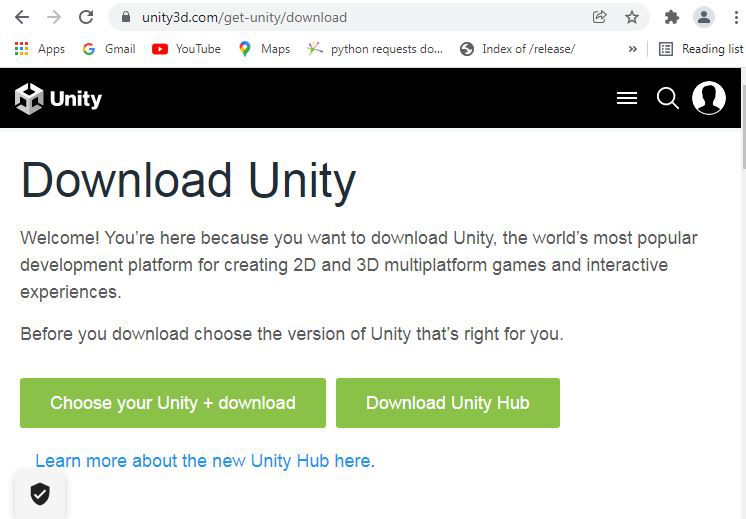
* Feedback and Iteration:

Encourage user feedback and iterate on your VR application to improve the user experience and fix any issues.

Remember that VR development can be challenging, but with patience, practice, and a good understanding of the documentation, you can create exciting and immersive virtual reality experiences.

**STEP FOR INSTALLATION** :

Step 1: Visit the official Unity Website and download the installer using any web browser.

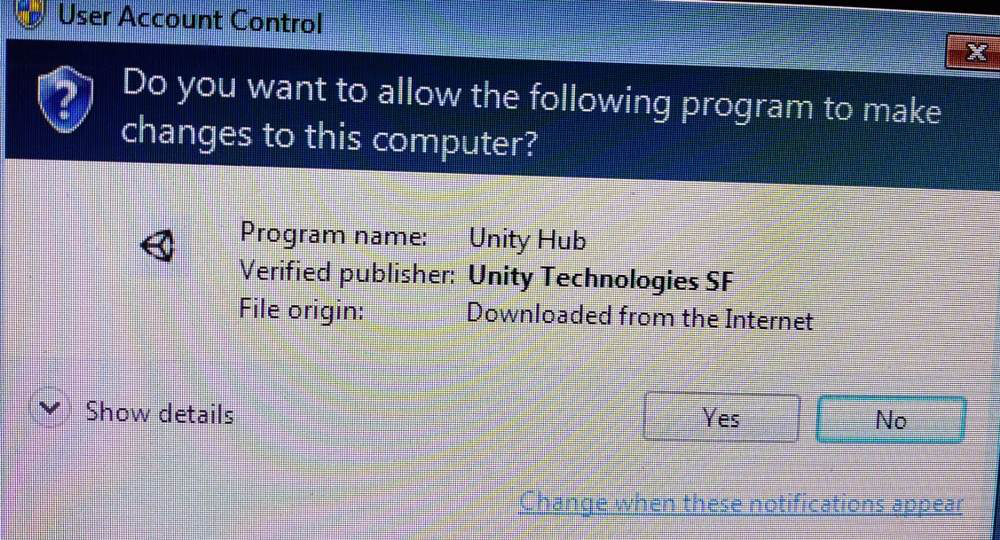


Step 2: Click on Download Unity Hub, downloading of the executable file will start shortly. It is a small 51.59 MB file that will hardly take a minute.

Step 3: Now check for the executable file in downloads in your system and run it.



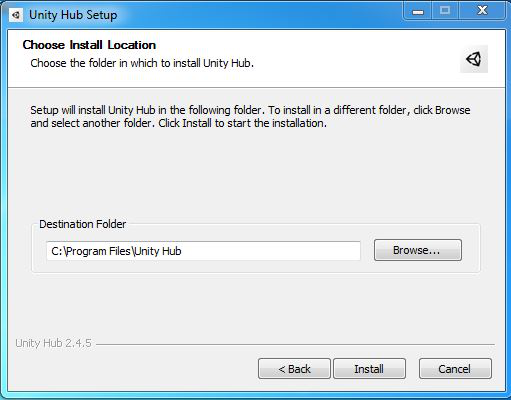
Step 4: It will prompt confirmation to make changes to your system. Click on Yes



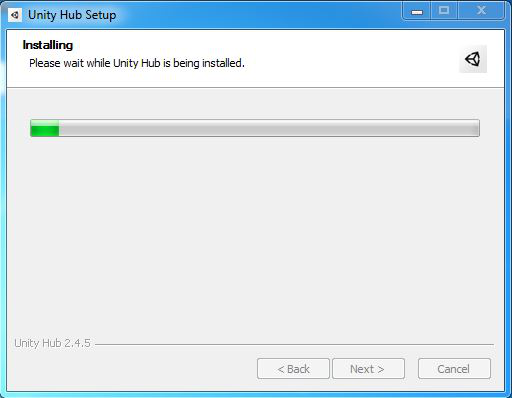
Step 5: The next screen will be of License Agreement, click on I Agree.



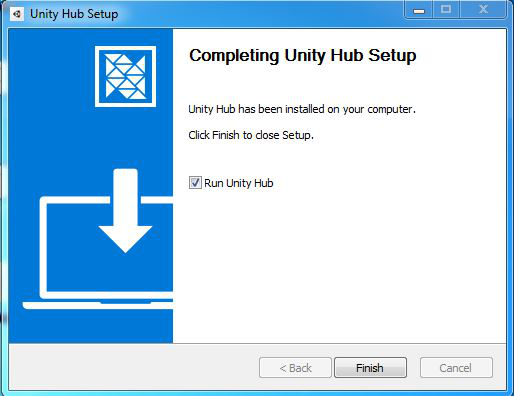
Step 6: The next screen will be of installing location so choose the drive which will have sufficient memory space for installation.



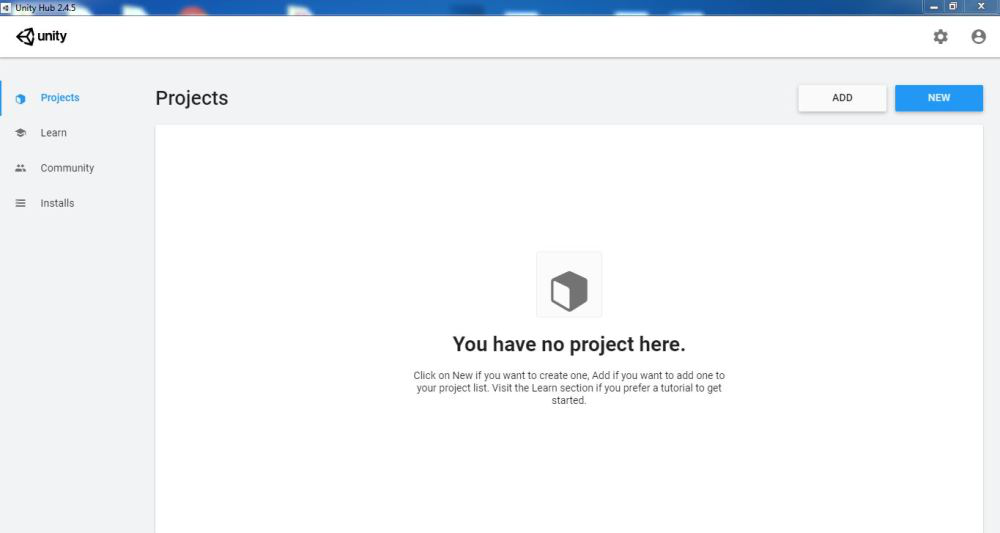
Step 7: After this installation process will start and will hardly take a minute to complete the installation.



Step 8: Click on Finish after the installation process is complete.

Step 9: Unity Hub is successfully installed on the system and an icon is created on the desktop.

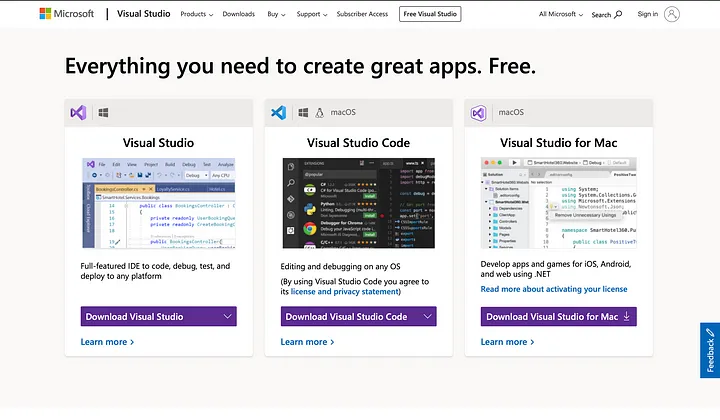
Step 10: Run the software and see the interface.



Downloading the correct Visual studio.

.

NOTE: The Professional and Enterprise editions are paid.



Depending on your platform, download the correct Visual Studio. Also, I wouldn’t recommend downloading ‘Visual Studio Code’ as it is just a code editor, not a fully integrated IDE unlike the Visual Studio

**CONCLUSION:**

In conclusion, the installation of Unity and Visual Studio, along with the setup of Unity for VR development and a comprehensive understanding of the associated documentation, form the foundation for a successful venture into the world of virtual reality. These steps not only provide the necessary tools and knowledge to create immersive VR experiences but also emphasize the importance of continuous learning and community engagement in this rapidly evolving field. By mastering these fundamentals, developers can embark on a journey to craft compelling and innovative VR applications that captivate audiences in the virtual realm.

|  |
| --- |
| Experiment No.2 |
| Demonstration of the working of HTC Vive, Google Cardboard, Google Daydream and Samsung gear VR |
| Date of Performance: |
| Date of Submission: |

**AIM:**

The aim of this demonstration is to showcase the working and capabilities of four distinct VR platforms: HTC Vive, Google Cardboard, Google Daydream, and Samsung Gear VR. This demonstration will provide an overview of the hardware, software, and user experience offered by each platform.

**OBJECTIVES:**

* Familiarization: Introduce participants to the key components and features of HTC Vive, Google Cardboard, Google Daydream, and Samsung Gear VR.
* Hardware Setup: Demonstrate how to set up each VR platform, including connecting sensors, headsets, and controllers (where applicable).
* Software Navigation: Show participants how to access and navigate the VR environments and apps specific to each platform.
* Interaction and Controls: Explain how users interact with the virtual world, including controllers, gestures, and gaze-based interactions.
* Content Showcase: Display a selection of VR content, games, or experiences available on each platform to highlight their unique offerings.
* Performance: Discuss the performance capabilities of each platform, emphasizing graphics quality, responsiveness, and tracking precision.
* Comfort and Ergonomics: Evaluate the comfort and ergonomics of the headsets, considering factors like weight, fit, and adjustability.
* Comparison: Provide a comparative analysis of the strengths and weaknesses of each VR platform in terms of immersion, accessibility, and content library.

**THEORY:**

Each of the mentioned VR platforms offers a distinct approach to virtual reality:

* HTC Vive: HTC Vive is a high-end VR system known for its room-scale tracking and precise motion controllers. It provides an immersive experience with a wide field of view and excellent tracking capabilities.



* Google Cardboard: Google Cardboard is an accessible and affordable VR option that relies on a smartphone for rendering. It offers a basic VR experience with limited interactivity, making it an entry-level option for VR exploration.



* Google Daydream: Google Daydream is a mobile VR platform that offers a more advanced and comfortable VR experience than Cardboard. It features a dedicated headset and controller, enhancing interactivity and content quality.



* Samsung Gear VR: Samsung Gear VR is another mobile VR platform that pairs with compatible Samsung smartphones. It provides a comfortable and user-friendly VR experience with a wide range of apps and games.



**CONCLUSION:**

In conclusion, this demonstration serves as an informative exploration of four diverse VR platforms. HTC Vive excels in providing high-fidelity, immersive experiences, particularly for gaming and interactive simulations. Google Cardboard offers a cost-effective introduction to VR, while Google Daydream enhances the experience with better hardware and interaction options. Samsung Gear VR, designed for Samsung smartphone users, balances comfort and usability for a wide range of VR applications. Understanding the unique features and capabilities of each platform helps users choose the VR experience that best aligns with their preferences and requirements.

|  |
| --- |
| Experiment No.3 |
| Develop a scene in Unity that includes:  i. a cube, plane and sphere, apply transformations on the 3 game objects.  ii. add a video and audio source |
| Date of Performance: |
| Date of Submission: |

**AIM:**

Develop a scene in Unity that includes:

1. a cube, plane and sphere, apply transformations on the 3 game objects.
2. add a video and audio source

**OBJECTIVES:**

* Game Object Setup and Transformation: Add cube, sphere, and plane to the scene, apply transformations (position, rotation, scale).
* Video Source Integration: Create a "VideoPlayer" GameObject, add Video Player component, and play a specified video clip within the scene.
* Audio Source Integration: Attach Audio Source component to a game object, play an audio clip, and configure audio settings.
* Camera and Lighting Setup (Optional): Position camera for scene capture, add lighting for visual quality.
* Testing and Interaction (Optional): Verify functionality, add user interactions, scripting if needed.
* Documentation and Presentation: Document the process and create a presentation.
* Optimization and Refinement (Optional): Optimize for performance and refine aesthetics based on feedback.

**THEORY:**

* Unity Game Objects and Transformations:

Unity is a popular game development engine that uses GameObjects as the fundamental building blocks of a scene.

GameObjects represent entities in the scene, and they can be 3D models, lights, cameras, or any other element.

Transformations in Unity involve manipulating a GameObject's position, rotation, and scale to place and orient it as needed within the 3D world.

To apply transformations, you use the Transform component of a GameObject. You can manipulate this component directly in the Unity Inspector or programmatically in C# scripts.

Common transformations include:

Position: Adjusts the object's location in the 3D space (X, Y, Z coordinates).

Rotation: Changes the object's orientation (Euler angles or Quaternion).

Scale: Modifies the object's size.



* Adding Video and Audio Sources:

Video Source Integration:

Unity provides the Video Player component for playing video content within a scene.

To integrate video:

Create an empty GameObject (e.g., "VideoPlayer") to serve as the video player.

Attach the Video Player component to this GameObject.

Import video clips into your Unity project or specify URLs for online videos.

Assign the video clip to the Video Player component.

Configure additional settings, such as video playback speed, loop behavior, and rendering mode.

Create a UI element or texture to display the video content, and set it as the target for the Video Player component.

Audio Source Integration:

Unity uses the Audio Source component to play audio content, including music, sound effects, and voiceovers.

To integrate audio:

Attach the Audio Source component to a GameObject (e.g., a cube or any object that should emit sound).

Import or specify audio clips to be played within the scene.

Assign the audio clip to the Audio Source component.

Configure audio settings such as volume, spatial blend, and 3D sound settings (if needed).

* Use C# scripts to control audio playback, such as triggering audio on specific events or in response to user interactions.
* Ensure that the video and audio formats you use are compatible with Unity.
* Optimize video and audio assets for performance and quality.
* Consider using UI elements, textures, or materials to display video content in the scene.

Use Unity's AudioSource and VideoPlayer documentation for more details on configuration and scripting.

By mastering these foundational concepts in Unity, you can develop a scene that includes 3D objects with transformations, video playback, and audio integration to create engaging and interactive experiences for your users.

**OUTPUT:**

**CONCLUSION:**

|  |
| --- |
| Experiment No.4 |
| Develop a scene in Unity that includes a cube, plane and sphere. Create a new material and texture separately for three Game objects. Change the color, material and texture of each Game object separately in the scene. Write a C# program in visual studio to change the color and material/texture of the game objects dynamically on button click. |
| Date of Performance: |
| Date of Submission: |

**AIM: -**

Develop a scene in Unity that includes a cube, plane and sphere. Create a new material and texture separately for three Game objects. Change the color, material and texture of each Game object separately in the scene. Write a C# program in visual studio to change the color and material/texture of the game objects dynamically on button click.

**OBJECTIVES: -**

The objectives for this Unity project are to create a scene with cube, plane, and sphere objects, each with unique materials and textures. The goal is to dynamically change the color, material, and texture of these objects individually upon clicking a button, achieved through a C# program. Key steps include scene setup, material and texture creation, assigning them to objects, implementing color changes, coding in Visual Studio, creating a user interface button, rigorous testing, and proper documentation. Successfully completing these objectives will enhance proficiency in Unity, C# programming, and interactive content development, fostering a well-rounded skill set in game development.

**THEORY: -**

This Unity project aims to create an interactive 3D scene comprising cube, plane, and sphere objects.

1. Scene Composition:
   1. Unity Scene: Unity scenes serve as the primary environment where game objects and assets are organized and interact. In this project, you'll create a new Unity scene to build your 3D environment.
   2. GameObjects: GameObjects are the fundamental building blocks of Unity scenes. You'll add three types of GameObjects – cube, plane, and sphere – to your scene to create the visual elements.
2. Material and Texture Creation:
   1. Materials: In Unity, materials define how an object appears by specifying its color, transparency, and how it reacts to lighting. You'll create separate materials for the cube, plane, and sphere to give them unique visual properties.
   2. Textures: Textures are 2D images applied to materials to add surface details, patterns, or realism. You'll create textures to enhance the appearance of your objects.
3. Material and Texture Assignment:
   1. Material Assignment: Unity allows you to assign materials to GameObjects. You'll assign the previously created materials to their respective cube, plane, and sphere GameObjects.
   2. Texture Assignment: Similarly, you'll assign textures to the materials, ensuring that each object gets the desired visual texture.
4. Color Modification:
   1. Dynamic Color Change: Through C# scripting, you'll create a mechanism to change the color of each GameObject individually. This will involve modifying the material's color properties during runtime based on user interactions, such as button clicks.
5. Material/Texture Swapping:
   1. Dynamic Material/Texture Change: You'll use C# programming in Visual Studio to enable dynamic changes in the material or texture properties of the GameObjects. These changes will occur in response to specific user actions, such as clicking a button in the Unity scene.

Theoretical understanding and practical application of these objectives will equip developers with essential skills in Unity, 3D asset manipulation, material and texture management, C# scripting for interactivity, and user interface integration. Ultimately, this project contributes to proficiency in Unity game development and interactive content creation.

**CODE: -**

**OUTPUT: -**

**CONCLUSION: -**

.

|  |
| --- |
| Experiment No.5 |
| Develop a scene in Unity that includes a sphere and plane . Apply Rigid body component, material and Box collider to the game Objects. Write a C# program to grab and throw the sphere using the vr controller. |
| Date of Performance: |
| Date of Submission: |

**AIM: -**

Develop a scene in Unity that includes a sphere and plane . Apply Rigid body component, material and Box collider to the game Objects. Write a C# program to grab and throw the sphere using the vr controller.

**OBJECTIVES: -**

This Unity project aims to create an interactive scene with a sphere and a plane, incorporating physics-based realism. It includes applying Rigidbody components to the objects for natural movement, assigning distinct materials for visual differentiation, and implementing Box Colliders for accurate collision handling. The primary objective is to enable interaction with a VR controller through a C# program, allowing users to grab and throw the sphere realistically. Extensive testing ensures a responsive and immersive VR experience, while documentation ensures the project's reproducibility and future development. This project provides practical experience in VR controller integration, physics simulation, and scripting, enriching the skill set for immersive VR development.

**THEORY: -**

The theory behind this Unity project involves several key aspects of game development and VR interaction:

1. Scene Creation:
   1. Unity Scene: Unity provides a platform for creating 3D environments and simulations. In this project, a scene is created to host the interactive sphere and plane GameObjects.
2. Physics Simulation:
   1. Rigidbody Component: Rigidbody is a Unity component that adds physics simulation to GameObjects. It enables realistic behaviors like gravity, collision detection, and dynamic movement. Applying Rigidbody to the sphere and plane ensures they respond to physics forces accurately.
3. Material and Visuals:
   1. Materials: Materials define the visual properties of objects, including their color, texture, and transparency. Applying materials to GameObjects allows for customization of their appearance.
4. Collision Detection:
   1. Box Collider: Box Collider is a type of Collider component in Unity used for defining the shape and boundaries of 3D objects. Applying Box Colliders to the sphere and plane ensures that they can interact with each other and other objects in the scene.
5. VR Controller Interaction:
   1. C# Scripting: Unity allows developers to use C# scripts to create interactive behaviors. In this project, a C# script is written to interface with a VR controller.
   2. VR Controller Input: VR controllers have input sensors and buttons that can detect user interactions. The script interprets these inputs to enable the user to grab and release the sphere.
   3. Physics-Based Throwing: The C# script applies forces to the sphere based on the controller's movement, simulating the action of grabbing and throwing it within the VR environment.

In summary, this project combines Unity's physics simulation, material and collider components, C# scripting, and VR controller integration to create an engaging VR experience where users can interact with and throw a sphere within a virtual environment. These skills are fundamental for developers interested in VR game development and interactive simulations.

**CODE: -**

**OUTPUT:-**

**CONCLUSION: -**

|  |
| --- |
| Experiment No.6 |
| Develop a simple UI(User interface ) menu with images, canvas, sprites and button. Write a C# program to interact with UI menu through VR trigger button such that on each successful trigger interaction displays a score on scene. |
| Date of Performance: |
| Date of Submission: |

**AIM: -**

Develop a simple UI(User interface ) menu with images, canvas, sprites and buttons. Write a C# program to interact with UI menu through VR trigger button such that on each successful trigger interaction displays a score on scene .

**OBJECTIVES: -**

This Unity project aimed to create an interactive VR user interface (UI) featuring images, buttons, and a scoring system. It involved setting up a Canvas to host UI elements, scripting in C# to enable interactions via a VR trigger button, and dynamically displaying scores on the scene upon successful triggers. The project emphasized the integration of VR technology with Unity's UI capabilities, enhancing user engagement. Rigorous testing ensured seamless VR interactions, while code optimization prioritized performance. Comprehensive documentation was provided to aid future reference and collaboration. In summary, this project equipped developers with valuable skills in VR UI design and scripting, enriching their capabilities in VR application development.

**THEORY: -**

The theory behind developing a UI menu in Unity with images, canvas, sprites, buttons, and integrating it with a VR trigger button to display a score upon interaction involves several key concepts:

1. Unity UI Components:
   1. Canvas: A Canvas is a fundamental UI component in Unity. It acts as a container for all other UI elements and provides a plane on which UI elements are rendered.
   2. Images: Images are UI elements used to display graphics or visual elements on the Canvas. They can represent background images, icons, or any visual content.
   3. Sprites: Sprites are 2D images or graphics used in Unity to represent objects or elements within the game or UI.
   4. Buttons: Buttons are interactive UI elements that users can click or, in the case of VR, trigger interactions with.
2. C# Scripting:
   1. C# Programming: Unity uses C# scripting to create interactive behaviors. In this context, C# scripts are written to handle the interactions between the VR trigger button and the UI elements.
   2. VR Integration: Integration with VR hardware, such as Oculus or HTC Vive, involves capturing input events like trigger presses.
3. Interactivity:
   1. VR Trigger Interaction: The VR trigger button serves as the input mechanism for interacting with the UI menu. When the trigger is pressed or activated, it initiates an interaction event.
   2. Score Display: Upon a successful interaction event, the C# script updates and displays a score on the UI Canvas within the scene.
4. Testing and Optimization:
   1. Testing: Rigorous testing is essential to ensure that the VR trigger interactions function correctly, and the score display updates accurately.
   2. Performance Optimization: Optimization of code and UI elements ensures smooth and responsive interactions within the VR environment.
5. Documentation:
   1. Code Documentation: Comprehensive documentation of the C# scripts and UI setup is critical for understanding and maintaining the project and for future reference and collaboration.

In summary, this project combines Unity's UI components, C# scripting, and VR integration to create an interactive UI menu. It demonstrates the potential of VR technology to enhance user engagement by allowing users to interact with UI elements using a VR trigger button and dynamically display scores as a result of successful interactions. This project enriches developers' skills in VR application development and UI design for immersive experiences.

**CODE: -**

**OUTPUT: -**

**CONCLUSION: -**