Implementation of a sample application for the Augmented Reality glasses Microsoft HoloLens

Author: Xiangyu Tong
Supervisors:
Dr. Thomas Kronfeld

Technische Universität Chemnitz

xiangyutong4cf@gmail.com

September 2, 2021

Content

Motivation

Microsoft HoloLens is a mixed reality eyewear that allows the user to display interactive 3D projections in the direct environment with the support of a Natural User Interface. The device uses sensual and natural interface commands like gaze, gesture and voice. For that, this project aimed to give a broad audience access to this technology. Besides, it is of interest to develop a simple and intuitive application for HoloLens and test its performance.

Task

In order to take advantage of the natural interface commands of HoloLens, such as gaze, gesture, a “Whac-A-Mole” like application should be developed within the context of the research project.

So the goal is the development of a game with the following features:

1. Determine a flat region in front of the user
2. Display virtual 3D model on top of the determined region, at random locations for a random
3. If the user reaches or taps on one of these models, the model disappears and an internal counter is increased.
4. After 90 seconds the game is over and the score of the user should be displayed
5. A restart can be triggered with the tap gesture. The environment for testing along with the Microsoft HoloLens device will be provided by the supervisor.

Prerequisites

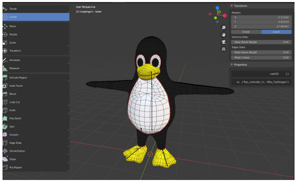
To start implement the game, the following tools are necessary:

- Visual Studio 2019
- Unity 2018.4.x or later
- Windows SDK 18362+
- Mixed Reality Toolkit

The HoloLens Emulator is recommended to use for pre-test of the application. The requirement of Hard-ware is already talked in the report of this project.

3D-model

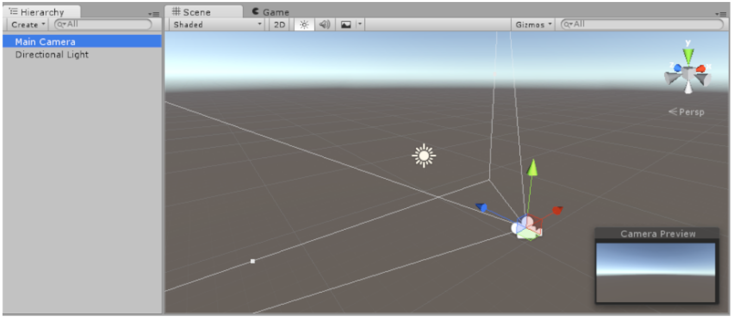
Here in this application the most well-known penguin “Tux” is applied as “mole”



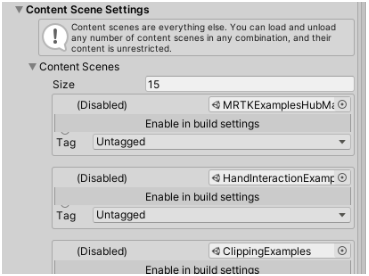
3D-models can be downloaded online from model-shop such as Sketchup Free, CGTrader. Also, 3D-models can be created using some tools like Blender.

Scene

Scenes are where the user works with content in Unity. They are assets that contain all or part of a game or application. For a simple game, a single scene might be enough, while for a more complex game, user might use one scene per level. The number of scene in a project depends on the complexity of the application.[1]



Scene Management



Content Scene Settings

Content scenes are everything else. You can load and unload any number of content scenes in any combination, and their content is unrestricted.

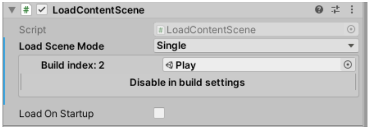
Content Scenes

Size: 15

(Disabled) MRTKExamplesHubMa
Enable in build settings
Tag: Untagged

(Disabled) HandInteractionExamp
Enable in build settings
Tag: Untagged

(Disabled) ClippingExamples
Enable in build settings

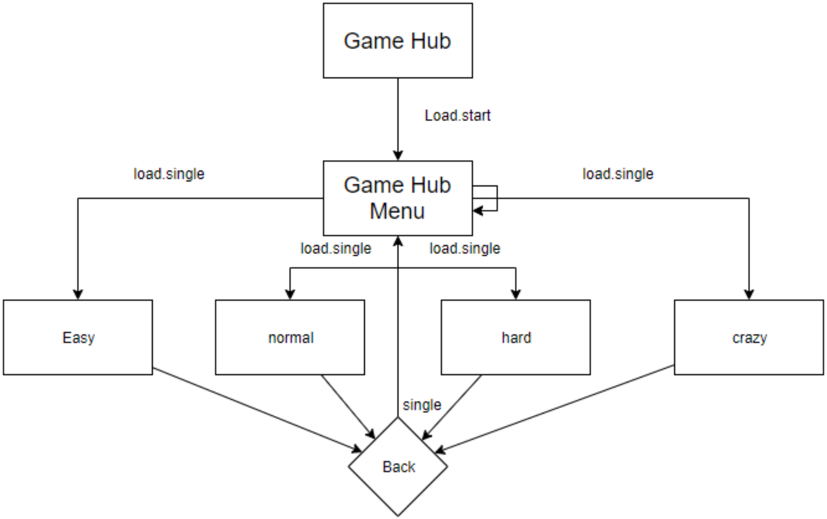


LoadContentScene

Script: LoadContentScene
Load Scene Mode: Single
Build index: 2 — Play
Disable in build settings
Load On Startup: ☐

Multi Scene Editing allows you to have multiple scenes open in the editor simultaneously, and makes it easier to manage scenes at runtime. MRTK offers SceneSystem to manage and set scenes. All content load operations are asynchronous, and by default all content loading is additive. Single-Scene-Loading can be implement by changing the mode of Load-Scene. Manager and lighting scenes are never affected by content loading operations.

Scene Management



Game Object

In a scene of play, there are two kinds of object:

Concrete Object:

- Camera
- Tuxes
- Cursor
- Texts (UI)

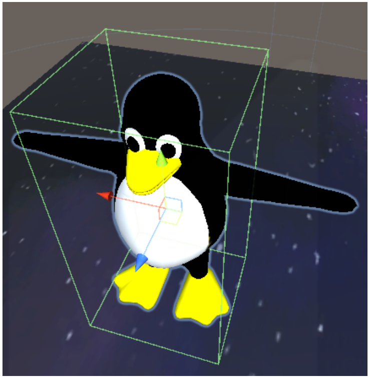
Abstract Object:

- Game Control

According to the game logic of this project, each Object is attached to one or more corresponding c# scripts

Game Logic and Methods

Collider

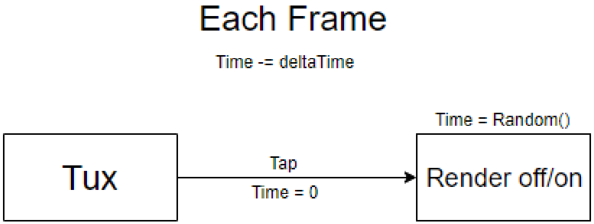


To implement the Tap-Gesture-Interaction, a collider is attached to each Tux. Collider defines the valid area involves interaction in gaze and tap-gesture. Hence, the size of collider depends on the specific situation. Furthermore, a group of colliders can be used for one model, so that various physical effect can be implemented in different sub-colliders. Interactive.

Motion

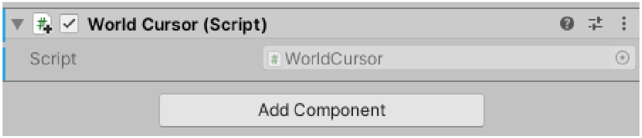
For developing a "Wahc-A-Mole" like application, the "Tuxes" should be displayed at random locations for a random amount of time. Thus, the author adds a script- component for each Tux. The scripts initializes objects and updates the behavior of objects once per frame:

The basic idea from the author is that each Tux has its own "remaining time" and disappears by Tap-Gesture or remaining time reduces to 0s.



Gaze and Cursor

One of the interactivities to holograms is Gaze. Gaze is exact the user's attention, where they'are looking in the world around them. By using Gaze the user can determine which hologram he or she intends to interact with, that is, when the player target an object, the cursor occurs on the surface of the object and then moves along with the ray of the player. First, a cursor-asset is added to the scene by draging it up to the Hierarchy pane at the root level. Then, to make the cursor move around every frame, a script is attached to it.

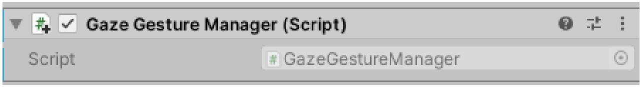


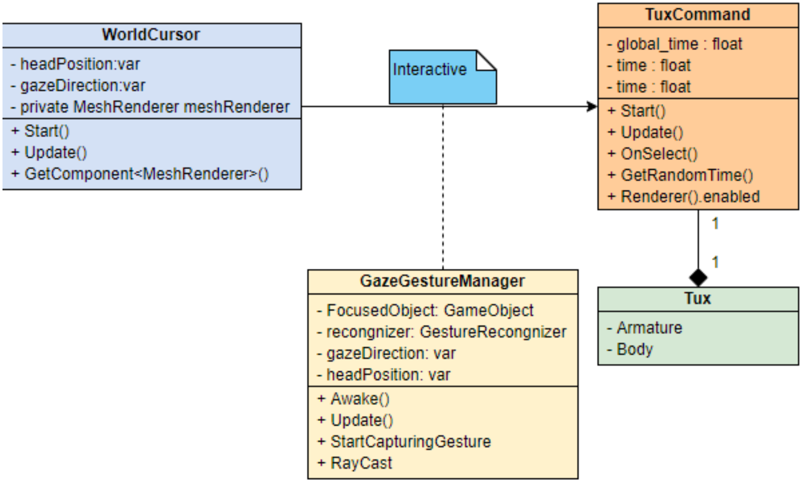
World Cursor (Script)

Script: WorldCursor

Gesture:Tap

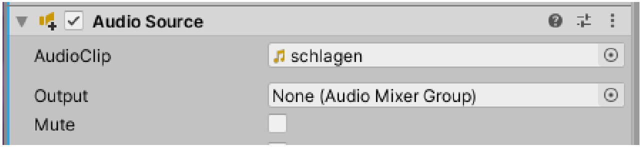
A script supporting for gestures is attached to each Tux. It manage gaze and gesture with the help of the two scripts above. The main idea is that, in each frame it first checks if one object is focused by gaze. If yes, then it checks and recognizes the gesture by the player and send the tap-message to the script of Tux-command.


Gaze Gesture Manager (Script)
Script GazeGestureManager


WorldCursor
- headPosition:var
- gazeDirection:var
- private MeshRenderer meshRenderer
+ Start()
+ Update()
+ GetComponent<MeshRenderer>()
Interactive
TuxCommand
- global_time : float
- time : float
- time : float
+ Start()
+ Update()
+ OnSelect()
+ GetRandomTime()
+ Renderer().enabled
1
1
Tux
- Armature
- Body
GazeGestureManager
- FocusedObject: GameObject
- recongnizer: GestureRecongnizer
- gazeDirection: var
- headPosition: var
+ Awake()
+ Update()
+ StartCapturingGesture
+ RayCast

Audio Effect

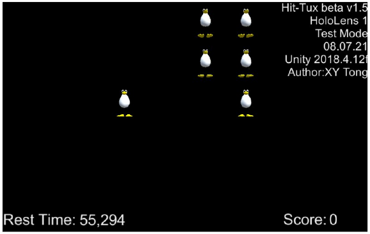
If one Tux was selected and tapped, the player wish to get a feedback so that he or she know the Interaction with the Tux is successful.



Audio Source

AudioClip	schlagen
Output	None (Audio Mixer Group)
Mute	☐

Visualization of the application in HoloLens



Once the project and corresponding scene are properly configured, the first thing to add is a Canvas object to the scene to use as a surface to write on. By default, the Canvas will be locked to the world space. To configure a camera-locked view, select the Canvas and examine its properties in the inspector window.

Then in the render mode filed of the Canvas, select Screen Space - Camera in the drop down menu and drag the Main Camera into the Render Camera filed of the Canvas. This tells the Canvas which camera perspective it is locked to. As a result, the terminator vision will be permanently locked to the display of HoloLens.

References

MRTK-Doc
MRTK-Unity-Documentation

Microsoft HoloLens-Tutorial
HoloLens-Documentation

The End