A Toolkit to Improve Augmented Reality Game Development Workflow

Devi Selvam[1], Ashwin Balaji.S[2], Mohammed Imran.H[3], Monichandru.M[4]
Department of Computer Science and Engineering
[1] Assistant Professor(SG), Sri Shakthi Institute of Engineering and Technology, Coimbatore
[2,3,4] B.E Scholar, Sri Shakthi Institute of Engineering and Technology, Coimbatore

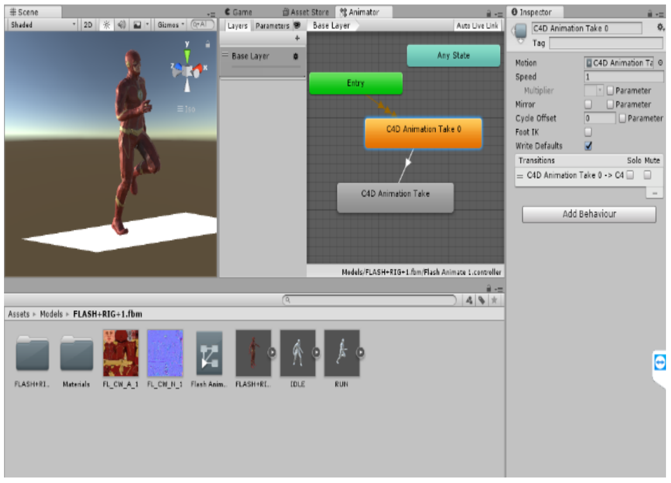
Abstract- *There are many tools available to create augmented reality content for different platforms. However, there is no tool which speeds up development of augmented reality games for the Unity Engine. Our aim is to create a toolkit for use within the Unity platform which provides developers the basic functionality and assets required to quickly prototype their ideas and test their concepts. This saves a lot of time for developers who are looking to quickly make changes to their projects and test them in real time.*

Keywords- augmented reality, gameplay system, development toolkit, geo-fence, plugin

I. INTRODUCTION

Augmented reality games have created a new trend in the gaming industry. In recent times, many indie developers and leading developers who create AAA games have experimented with Augmented Reality technologies in their games. Indie developers who are limited by budget and time, do not have enough resources at their disposal to prototype and work as fast as the leaders in the industry. This is because not much attention has been paid to the creation of tools which speed up development for open source engines such as Unity.

Games such as Pokemon Go have had huge success in the past few years and have set a benchmark for other games.



Our solution to this problem is to create a plugin for the Unity Engine which provides the necessary tools and assets for creating a prototype of an idea as quickly as possible. This can be done by providing developers a Unity package which serves as a toolkit as well as gameplay system within the Unity Editor.

II. REASON

One key issue with the existing workflow is that the animations must be separate from the 3d model or asset being used. This leads to confusions, especially when working with multiple models in the project as the raw animation file is not enough to identify to which model it belongs.

The goal is to create a unified interface from where all animations can be mapped to the various models.

There is also no native support for location based tracking in the existing system which many developers require. This is implemented by allowing developers to specify triangulation coordinates in the form of latitude and longitude. This information is then used to create a geofence within which the different objects are set to spawn based on a trigger.

This trigger could be anything from a simple animation to a sound played around the device.

Once the geofence is created, the scene is populated with the required models.

III. WORKING

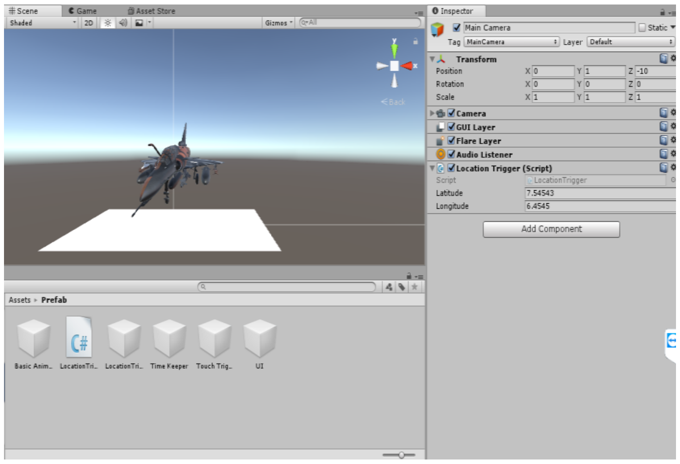
The tracking of images is done by using a target image for reference. This image must be of high resolution with a good colour depth.

By having multiple reference or tracking points, the image is tracked almost instantly.

This is indicated by a star rating, ranging from 0 to 5, zero being the lowest quality and 5 being the highest quality.

The same holds true for multiple targets also, where in a single scene, multiple images can be set as a target for a single canvas, leading to a multiple choice scenario while tracking.

Multiple targets can be very useful when designing scenes where the targets can be of different types.



Main Camera
Tag MainCamera Layer Default
Transform
Position X 0 Y 1 Z -10
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Camera
GUI Layer
Flare Layer
Audio Listener
Location Trigger (Script)
Script LocationTrigger
Latitude 7.54543
Longitude 6.4545

IV. ESSENTIALS

Object tracking is another form of motion tracking where a real life object can be used as a reference for a model or a scene.

For the user interface, we use a canvas template which can be customized as per the developers' preferences. This is a very generalized solution to project a UI within the game space. Animations to the UI elements are also supported.

We need a signal or a trigger to start the next event in the scene or jump to the next scene.

This is handled using a dedicated component called the TimeKeeper.

The TimeKeeper is basically a C# script integrated with the Unity UI namespace, which allows it to be projected as a tool within the Unity Editor.

For the purpose of instructing the gamer about the upcoming scene, an option of intermediate scenes is also provided which can also be customized to create a unique experience.

V. ADDITIONAL FEATURES

Touch Trigger is also another feature which allows developers to treat a touch on the object as a trigger to proceed to the next game event or transition to a different scene and so on.

Location based tracking is one of the major features, which will help developers place their models in real life locations right from the Unity plug-in by specifying a latitude and longitude within the Editor.

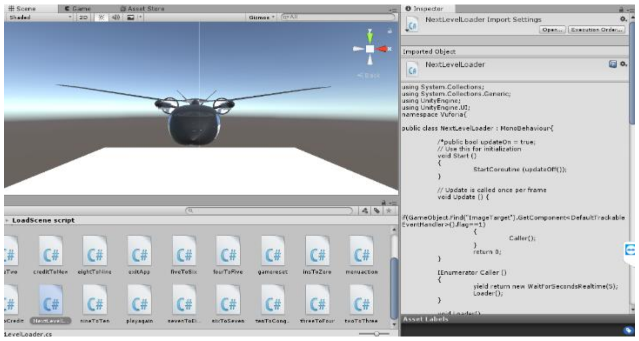
Three sets of latitudes and longitudes are needed to form a geofence. This virtual geofence is used to set the boundary or limit of the current game scene or level.

VI. COMPOSITION

All the components designed for the toolkit are basically C# scripts written to be extensible and compatible with the Unity Editor and its respective APIs.

This will be aggregated as a Unity package and released on the public domain as open source software under the GNU General Public License (GPL).

Any major revisions to the toolkit will be handled using a private Git repository (to be created during release phase of the project).



NextLevelLoader Import Settings
Imported Object
NextLevelLoader

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
namespace Vuforia{

public class NextLevelLoader : MonoBehaviour{

        /*public bool updateOn = true;
        // Use this for initialization
        void Start ()
        {
                StartCoroutine (updateOff());
        }

        // Update is called once per frame
        void Update () {


if(GameObject.Find("ImageTarget").GetComponent<DefaultTrackable
EventHandler>().flag==1)
                {
                        Caller();
                }
                return 0;
        }

        IEnumerator Caller ()
        {
                yield return new WaitForSecondsRealtime(5);
                Loader();
        }
```

VII. CONCLUSION

This project is expected to save countless development hours, especially for indie developers who would benefit greatly from having a toolkit which provides commonly reused components out of the box and supports location based augmented reality tracking and geo-fencing.

REFERENCES

[1] https://developer.android.com/develop/index.html
[2] https://en.wikipedia.org/wiki/Augmented_reality
[3] https://unity3d.com/learn/tutorials/topics/developer-advice/how-start-your-game-development
[4] http://www.engpaper.com/augmented-reality-2016.htm