Changing Space Shooter game into AR Space Shooter Game in 30 minutes

Project preparation

1. Create new empty project in Unity 3D

2. Download Space Shooter game from Asset Store
   https://www.assetstore.unity3d.com/en/#!/content/13866

3. Import Space Shooter game to your project.

4. Download latest Vuforia SDK for Unity from https://developer.vuforia.com/resources/sdk/unity and import to your project

5. Download LiGaBo SDK for Unity and Vuforia from http://developer.livegameboard.com/ and import it to your project

Enabling Augmented Reality in the game

1. Open scene Done/Done_scenes/Done_main

2. Go to Assets/Plugins/Ligabo/Prefabs and drag the LigaboCamera prefab into your scene and set its X and Z to 0 so it nicely points down to our scene. Delete or disable original Main Camera. LigaboCamera should be the main camera in the project.

3. Add LigaboTarget located in the same folder as LigaboCamera. Drag LigaboTarget to Hierarchy view. Set its position to 0. When you have LigaboTarget object selected, in inspector on Image Target Behavior script set drop down Data to empty and then back to Ligabo (needed just due to some funny state of this after putting prefab into scene)

4. Create empty GameObject, rename it to Content. And set its position to 0 and scale to 1.

5. In the hierarchy view, select all game objects that represent game scene which you want to be Augmented and drag them under Content object. These are Done_player, Starfield, Background and Boundary. We leave out Display text as this is GUI element and it does not batter where it is. We leave out also Lighting because in AR only camera moves around the target game object but object itself is static so lighting can stay where it is.

As you can see there are some game objects which are not aligned with Ligabo Target (zero plane). For example stars in space are offset vertically to create effect of being in distance. How ever this does only work when camera is pointing in right angle down to game scene. In Augmented reality camera is controlled by player and is dynamically moving. This effect is not working anymore for this situation.

6. Select all game objects under Content which are offset and set their Y position to 0.

7. Now we need to match size of Live Game Board with the game scene. First lets rotate LigaboTarget around Y by -90 degrees so it is aligned by long side with long side of space plane. Then modify scale of LigaboTarget to match its width exactly with space plane. Scale should be about 21.

8. Now move Content object under LigaboTarget.

When you now start the game in editor, you should see feed from camera attached to your computer.

(You need Unity Pro for this. If you don't you'll need to build to your mobile device to test. In case camera feed won't show in editor player, check settings of web cam behavior script attached to LigaboCamera object. Make sure your camera is listed and selected). As you can see game scene is not visible at this stage. Take camera and point it to Live Game Board (if you do not have one printed yet, go to www.livegameboard.com/shop/ and download one for self print there and print it). Game scene shows up and is Augmented in camera feed sitting on the game board.

We have done basic set up of augmentation how ever the game need some special awareness of game board tracking state. When board is not being tracked game scene is hidden, it must be otherwise it would appear just hanging on one place when tracking is lost. However when game is not visible player cannot play it so the game must be properly paused until board is tracked again.

Making the game AR aware

1. Drag LiGaBo prefab located in the same folder as LigaboCamera to scene.

LiGaBo prefab has attached main LigaboState class which does provide information about tracking state. It also provide GUI layer with instructions for player to point device camera to Live Game Board when he is not doing so.

When game is started it normally shows some introduction screen or menu. Lets implement similar functionality here and show some game background and button to start game.

2. Open Done_GameController script attached to Game Controller game object in the scene. Add onGUI method to show button to start game.

   Add on top of the script

```
using Ligabo;
```

   then create OnGUI method with this content

```
void OnGUI(){

        float butWidth=300f;
        LigaboUtils.GUIScaleBegin();

        if (!started && LigaboState.status!=LigaboState.Status.Running){
                if (GUI.Button(new Rect(LigaboUtils.screenWidth/2-butWidth/2,
LigaboUtils.screenHeight/2-butWidth/5/2,butWidth,butWidth/5),"Start Game"))
                        LigaboState.status=LigaboState.Status.Running;
        }

        LigaboUtils.GUIScaleEnd();
}
```

3. We want to start game on button press not automatically so comment out his line in Start method

```
//StartCoroutine (SpawnWaves ());
```

4. In Done_GameController script create public variable to recognize current game state.

```
public bool started = false;
```

When you start the game now, you will get the button Start on your screen. When you press it Ligabo is set to state running which causes that you will get GUI info overlay telling you to point camera to live game board. This always shows up when Ligabo is running and board is not being tracked.
Point camera to live game board. Scene will show up and will start to augment. How ever the game does not start. We need to recognize when the game should be started, when it should be paused and resumed.

5. The game should start when Player pressed button start and when game board is being tracked. Lets add code to Update function to watch for this conditions and start game.

```
void Update () {
	if (!started && LigaboState.statusTracking==LigaboState.StatusTracking.Tracking) {
		StartGame();
	}
}
```

add method to start the game

```
void StartGame(){
	StartCoroutine (SpawnWaves ());
	LigaboState.status=LigaboState.Status.Running;
	started=true;

}
```

When you play the game, press start button and point camera to game board, game starts now. However there is no way to play the game at the moment as we have nothing to control the ship with. We'll create space ship controller a bit later. There is one more thing to do which is pausing the game when game board tracking is lost and resuming the game when it gets tracked again.

6. Add functionality of pausing and resuming to Done_GameController script

add local variable indicating pause status

```
public bool paused = false;
```

add pause and resume methods

```
void PauseGame(){
	Time.timeScale=0;
	paused=true;
}

void ResumeGame(){
	Time.timeScale=1;
	paused=false;
}
```

to Update() method add these two conditions to start and pause game when needed.

```
if (started && LigaboState.statusTracking!=LigaboState.StatusTracking.Tracking && !paused)
	PauseGame();

if (started && paused && LigaboState.statusTracking==LigaboState.StatusTracking.Tracking )
	ResumeGame();
```

to Start() method add set game time scale to 0 to have it stopped at beginning

```
Time.timeScale=0;
```

and to StartGame() method add set game time scale to 1 to start it

```
Time.timeScale=1;
```

Now when game is started and tracking of game board is lost, game pauses. When tracking is available again game is automatically resumed.

Adding character control

In Augmented reality playing mode specific character control is required. There are different control modes which are suitable for AR. In this particular case we'll let space ship to follow direction where camera is pointing to.

1. Select Done_player game object under Content and attach to it script Follow Camera located in Ligabo/Scripts folder.
2. Check move object automatically.
3. Update min movement value to 2 to reflect correctly scale of this game
4. Assign pointer texture to Target GUI texture so we will see where center of camera is pointing to

Start the game and ship should now follow center of the camera but should not exceed defined boundaries. Space ship should be firing so lets make this happen on touch. When screen is touched, space ship will fire.

5. Open script Done_PlayerController in Done/Done_scripts folder and in Update functions replace

   ```
   Input.GetButton("Fire1")
   ```

   with

   ```
   (Input.GetButton("Fire1") || Input.touchCount > 0)
   ```

   When you now touch the screen space ship will fire.

We are pretty much done now, however the game has some nice movement effects on space ship like rotating to fly direction and fire from engine. Will be nice to keep that in the game. So for this do following.

1. In inspector on Follow camera script uncheck Move Object Automatically. This will cause that space ship will not be moved by the script, but instead input axis information will be provided for other scripts to read it.
2. Lets change Done_PlayerController script on space ship game object to read movement input information from Follow camera script.

   On top of the script add

   ```
   using Ligabo;
   ```

   Create local variable

   ```
   private FollowCamera followCam;
   ```

   Add this to the beginning of `FixedUpdate()` method

   ```
   if (followCam==null)
     followCam=GetComponent<FollowCamera>();
   ```

   Replace these 2 lines in `FixedUpdate()`

   ```
   float moveHorizontal = Input.GetAxis ("Horizontal");
   float moveVertical = Input.GetAxis ("Vertical");
   ```

   with

   ```
   float moveHorizontal = followCam.moveHorizontal;
   float moveVertical = followCam.moveVertical;
   ```

Now the movement of the space ship keeps its effects.

Some little final modifications to disable tap firing when game is not running and restarting game by tap instead of pressing R button

in Done_GameController `Update() function replace`

```
Input.GetKeyDown (KeyCode.R)
```

with

```
Input.GetButton("Fire1") || Input.touchCount > 0
```

in Done_GameController SpawnWaves() replace

```
"Press 'R' for Restart"
```

with

```
"Tap to restart"
```

in `Done_PlayerController Update`() change condition

```
if (Input.GetButton("Fire1") && Time.time > nextFire)
```

with

```
if ((Input.GetButton("Fire1") || Input.touchCount>0) && Time.time > nextFire &&
Time.timeScale>0)
```

Now we are pretty much done.

Enjoy Augmented reality space shooter game.

You can try the game on Android device by installing it from http://www.livegameboard.com/apps/

Find more tutorials and developer support at http://developer.livegameboard.com/