Project brief

Ultimate Obby Creator Documentation

Owners Varun Aiyar and dylanhulin.focus@gmail.com

Welcome to the Documentation for the Ultimate Obby Creator [Remixable Framework]!

This documentation is structured in the following way:

- **General Overview:** An explanation of what the Framework provides right out of the box.
- **Manual:** How to use this framework to create a Competitive Obby game without touching a single line of code!
- **General QnA:** Answers to what we think would be common questions from creators using this framework that do NOT know how to code.
- **Technical Overview:** An explanation of what the Scripts inside the Framework do and ideas on how to expand on them.

General Overview

The Obby Framework comes with 4 Example Obby Stages.
It includes the following Script components:

- ObbyGameManager.ts
- EventRegistry.ts
- LocalComponents.ts
- ObbyUI.ts
- ObbyUIComponents.ts
- Persistence.ts
- QuestsManager.ts
- TriggerComponents.ts
- MoveAndRotate.ts (special thanks to Scott Slater (SlaterBBurn) for the MoveAndRotate script from his Animation package!)

It includes the following UI Components:

UI Components:

- StageHUD
- 

- StageProgressHUD
- 
- RaceTimerHUD
- 

- NotificationsHUD - Shows an animated Text Notification to the Player

You do not need to worry about what each of these scripts do to use this framework. In fact, you don't need to touch a single line of code!

(Note: These scripts will be explained in greater detail in the Technical Overview section)

Manual

How the Example Stages are created (and how you can create your own Stages!)

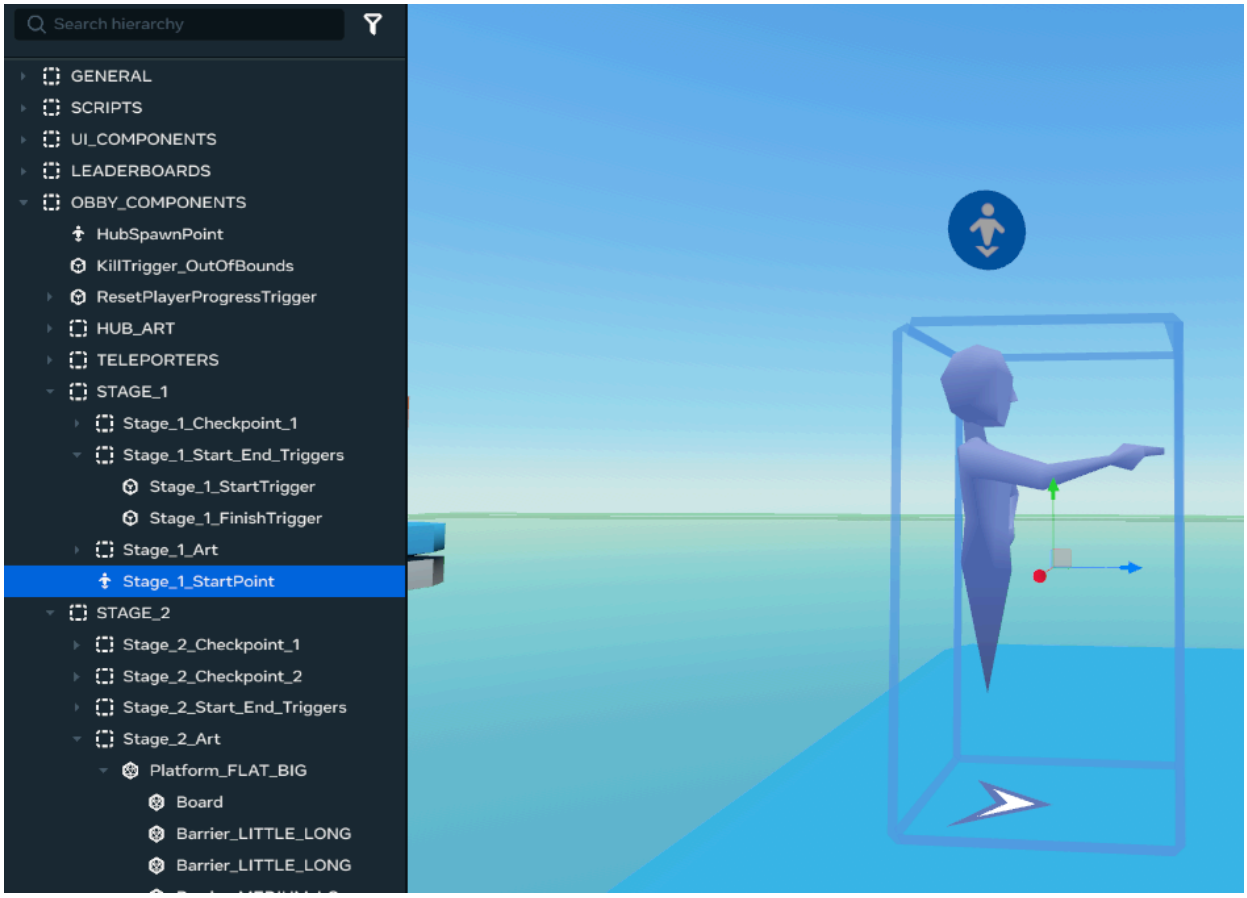
Each Stage in the framework needs the following things as its children:

- Start Spawn Point Gizmo
- Start Trigger
- Checkpoint Triggers (as many as you need)
- Finish Trigger

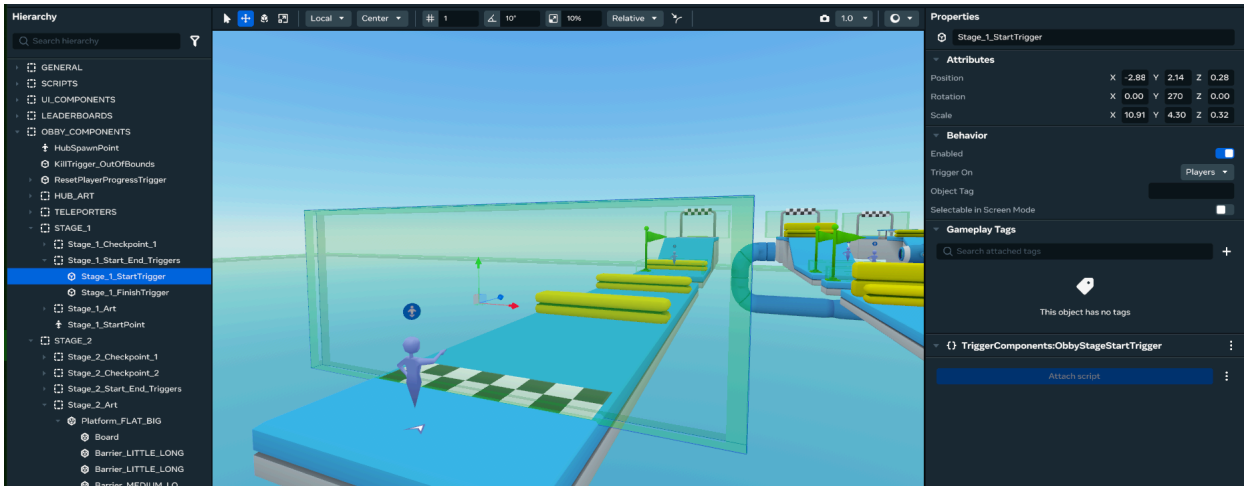
These elements are what define a Stage for the framework. Otherwise, it is not registered. We shall now explain how to add each of these elements for your own Stage.

(You can watch this video https://youtu.be/oAv4dAKNXBk for a quick timelapse of the process described in the following steps)

When you create an Empty Object in the editor as a Parent entity to start building your Stage, add a Spawn Point Gizmo that is named according to the following convention:
Stage_{yourStageIndex}_StartPoint
So if you are creating Stage 10, it would be Stage_10_StartPoint



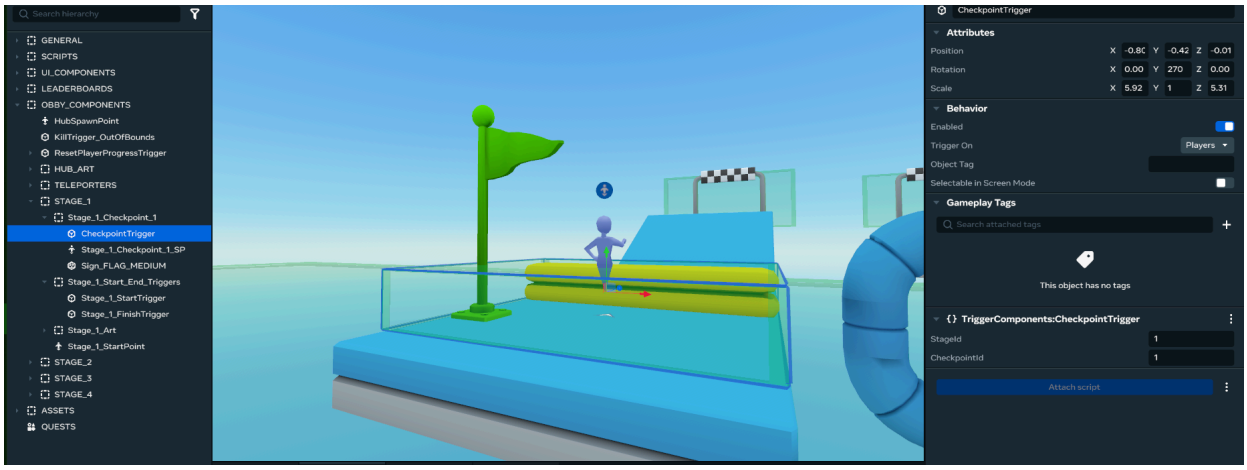
To create a Start Trigger, simply place a normal Trigger Gizmo wherever you want the starting line of your Stage to be, and attach the **TriggerComponent:ObbyStageStartTrigger** Script component to it.



To create a Checkpoint, first create an Empty Object and name it according to the following convention: **Stage_{stageIndex}_Checkpoint_{checkpointIndex}**

So if you are creating the 5th checkpoint for Stage 5, it would be Stage_5_Checkpoint_5

Inside this Empty but now renamed Object, place a simple Trigger Gizmo and rename it whatever you like. Let's say CheckpointTrigger for clarity's sake. Now, the important part is to attach the **TriggerComponent:CheckpointTrigger** Script component to it.



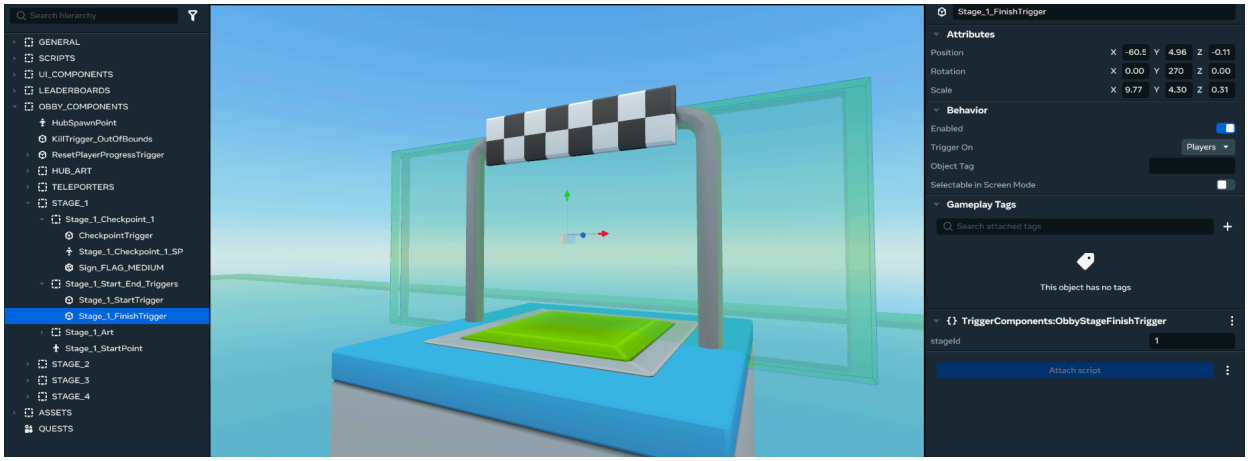
<u>Remember to set the correct Stage ID and Checkpoint ID in the Script's properties panel on the right side to match your Checkpoint's name!</u>

Now, you need a Spawn Point for the Checkpoint so the player automatically respawns at a checkpoint if they die after crossing it. Simply create a Spawn Point Gizmo inside your Checkpoint Parent and name it according to the convention:
Stage_{stageIndex}_Checkpoint_{checkpointIndex}_SP

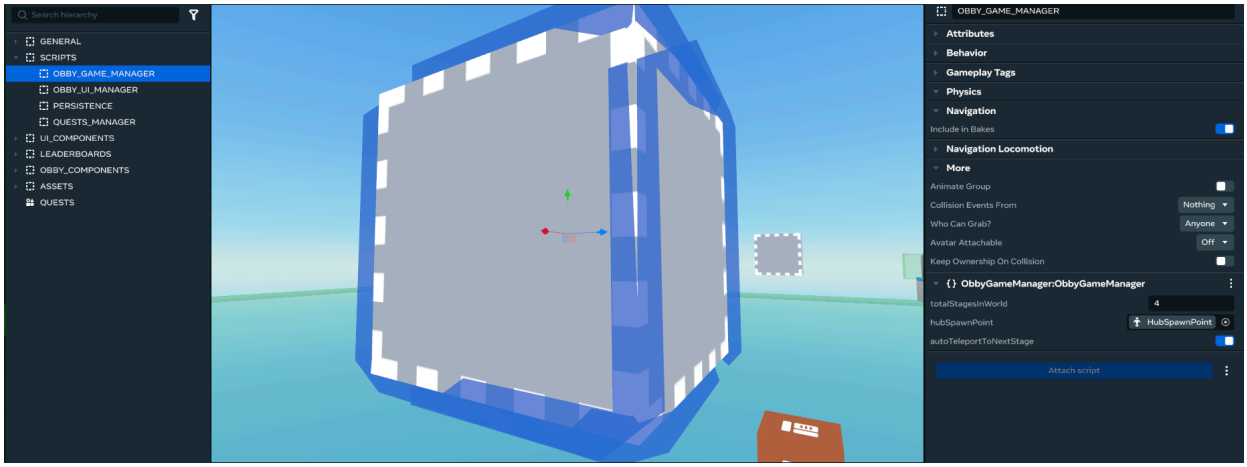
(Note that you must not stray away from these conventions!)

Now, you just need a Finish Trigger for your Stage. Simply add a new Trigger Gizmo under your Main Stage parent and attach the **TriggerComponent:ObbyStageFinishTrigger** Script component to it.

The important part here is to mention the correct stage ID for the Finish Trigger in the Properties panel on the right.



Lastly, but most importantly, select the OBBY_GAME_MANAGER under the SCRIPTS entity in the Hierarchy and change the totalStagesInWorld property to match the number of Stages you now have in the World.



And that's it!

Now you can add any meshes or other art elements you want to add to your Stage available from the framework's ASSETS entity (or add your own!)

Hit Preview! If you followed the above steps correctly to create your own Stage 5, you should be able to navigate to it upon completing Stage 4 using the Stage Navigation Buttons at the top of the UI.

General QnA

1) How do I add a Leaderboard for my new Stage?

A: Create a new Leaderboard from the Editor's Systems menu. If this is a Best Time leaderboard, make sure the order is Ascending (since lower times are better in this case), name it whatever you want. In the Persistence.ts script, add your leaderboard entry in these two places.

```
// Leaderboards
private static readonly LB_STAGE_1 = "Stage 1 Best Time";
private static readonly LB_STAGE_2 = "Stage 2 Best Time";
private static readonly LB_STAGE_3 = "Stage 3 Best Time";
private static readonly LB_STAGE_4 = "Stage 4 Best Time";
private static readonly LB_MOST_STAGE_COMPLETIONS = "Most Stage Completions";
```

(*Under these as LB_STAGE_5 or whatever you like*)

And in the switch statement here in the GetLeaderboardKey() function

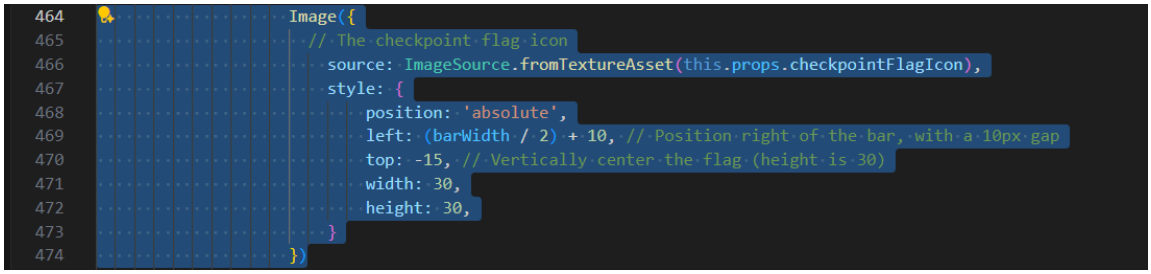
```
/** Gets the leaderboard key for the given stage */
private GetLeaderboardKey(stageId: number): string | null {
    switch (stageId) {
        case 1: return Persistence.LB_STAGE_1;
        case 2: return Persistence.LB_STAGE_2;
        case 3: return Persistence.LB_STAGE_3;
        case 4: return Persistence.LB_STAGE_4;
        default: return null;
    }
}
```

2) Is there a limit to the number of stages I can add in the game using this framework?

A: No, however, you need to stay within the recommended vertex count of Horizon Worlds to provide a smoother experience to players.

3) Is there a limit to the number of checkpoints I can add in the game using this framework?

A: No, however the CUI_StageProgressHUD will dynamically construct flag icons on the UI based on the number of Checkpoints you add in the stage. So, make sure to have a reasonable amount. Or if you want to get rid of the flag icon or make it smaller, edit the code in StageProgressHUD inside ObbyUIComponents.ts here



```
464     Image({
465         // The checkpoint flag icon
466         source: ImageSource.fromTextureAsset(this.props.checkpointFlagIcon),
467         style: {
468             position: 'absolute',
469             left: (barWidth / 2) + 10, // Position right of the bar, with a 10px gap
470             top: -15, // Vertically center the flag (height is 30)
471             width: 30,
472             height: 30,
473         }
474     })
```

4) How can I add a Quest?

A: Create the type of Quest you want and add it in the QuestsManager.ts script. You will need to write custom logic to update Quest Progress based on the type of Quest you want to add.

5) I don't want players to be automatically teleported to the next Stage. What should I do?

A: Select the OBBY_GAME_MANAGER entity in the Hierarchy and set the autoTeleportToNextStage boolean property to false. Now you either place your own Trigger Gizmo at the end of a Stage with a **TriggerComponents:ObbyStageTeleportTrigger** script attached with the correct stageID and use your own custom logic.

6) How do I create my own Traps?

A: Just design your own Trap asset and add a Trigger Gizmo with the **TriggerComponents:KillTrigger** script attached to it. The player will respawn to the Stage Start Point or Checkpoint based on their progress in that Stage.

Technical Overview

Back to the Scripts and UI Components! We shall now explain what each of these do.
Script components:

- ObbyGameManager.ts - This script performs several crucial tasks.
    - It dynamically finds all Start Spawn Points and Checkpoints for each Stage in the World based on a regex that looks for entities named based on the naming conventions we followed in the previous section of the documentation.
    - It keeps track of the stage any given player is actively playing
    - It handles teleporting Players to the Stages they select using the Stage Navigation Buttons in the Stage HUD and from ObbyStageTeleportTrigger calls
    - It calls functions from Persistence to save Player Progress at appropriate places in the gameplay loop and also updates Leaderboards and Quests via calls to Persistence and QuestsManager respectively
    - It keeps track of which stage any given player is actively playing.
    - It respawns the player at their most recent checkpoint on the stage they are actively playing
- EventRegistry.ts
    - This script is a simple registry of Network Events that are sent and received across the codebase
- LocalComponents.ts
    - This script is where all Local Scripts that run on the Player's device **locally,** should be defined. It contains the LocalRaceTimeManager, which itself is spawned from the ObbyUIManager for each player as an owner, so that their race time can be calculated locally.
- ObbyUI.ts
    - This script contains the ObbyUIManager class which holds references to all the UI Gizmos in the World and retrieves their UI Components on start() so that it can call functions on these UI Components
- ObbyUIComponents.ts
    - This script mainly contains the visual elements of the various UI in the game. It is where ALL the UI Components are defined.
- Persistence.ts

    - This script contains references to all Persistent Variables and Leaderboards in the World and functions to update them, that are called from ObbyGameManager
- QuestsManager.ts
    - This script contains references to all Quests in the World and a function to check whether those Quests were completed
- TriggerComponents.ts
    - If you've read this far, you know this Script contains all script components that are supposed to be attached to a Trigger Gizmo in the game World. The component names are self explanatory.
- MoveAndRotate.ts (special thanks to Scott Slater (SlaterBBurn) for the MoveAndRotate script from his Animation package!)
    - We recommend reading his package documentation Animation Pack | Devpost to understand how to use it.

UI Components:

- StageHUD
- 

- StageProgressHUD
- 
- RaceTimerHUD

- 

- NotificationsHUD - Shows an animated Text Notification to the Player