Tilt A Story: Developing Interactive Games for Kids using Unity and Playmaker

CS563
North Carolina State University – Raleigh

Team Members:

Ayush Gupta
Shifali Jain
Nishtha Garg
Sagar Manohar
Sreekanth Ramakrishnan

Milestones achieved

- Understanding the existing code and identify the refactorable components. ✅
- Build Menu development APIs.✅
- Build Environment development APIs. ✅
- Build Character development APIs. ✅
- Build Movement APIs. ✅
- Build Interaction APIs. ✅
- Build Sound integration APIs. ✅
- Build TiltAStory Game using the APIs ✅

User can customize the menu, environment, Character, Movement, Interaction, sound according to the game requirement using our custom scripts following the simple steps explained in our designer help document. We have created different scripts for all the parts mentioned above.

Menu: Menu can be customized using **LevelManager.cs** and **Playmaker FSMs**

Movement: The player movement is controlled using **PlayerController.cs**. This controls the accelerometer as well as they keyboard movements in the game.

Interaction: The pickups can also be controlled and customized using **PlayerController.cs**. To make an object rotate, user can attach 2 different scripts to the game: **Rotation.cs** and **Rotation1.cs**.

Sounds: Sounds are also controlled in the scripts. The designers can simply change them using the inspector in unity.

Environment: Changing game environment and background is easy and can be done using a simple series of steps as explained later in this project.

All the images, sounds and backgrounds are maintained in separate folders under the “Assets” folder in the project directory. The scripts are maintained under scripts folder.

We have also Re-created the old Tilt-a-Story game by using our API and the old assets including sounds, images and backgrounds.

Types of Games:

Three type of games with accelerometer based gameplays are currently supported by the Tilt A Story game design.

1. Follow a Path: Where the user has to follow a specific predefined path and reach the destination.
2. Select Objects: Where the user has to pick up specific objects from the group of objects.
3. Maze games: Where the user has to pick up specific objects in a maze.

Building a Game menu:

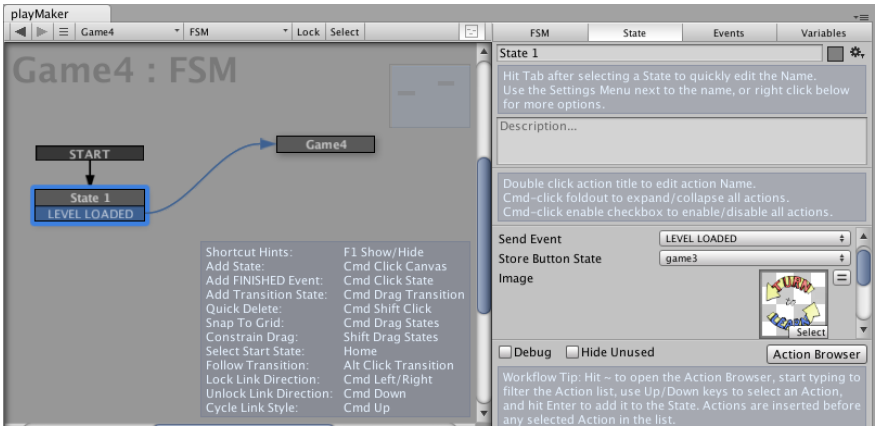
The generic game is built in such a way to support multiple stories inside the game. The current app as we see opens to menu screen with three icons. Each of these icons represents a separate type of game. Once you have developed a game, you can follow the steps below and easily add it to the existing game menu.

The scene used as the menu is present in /Assets/_Scenes/SelectGame

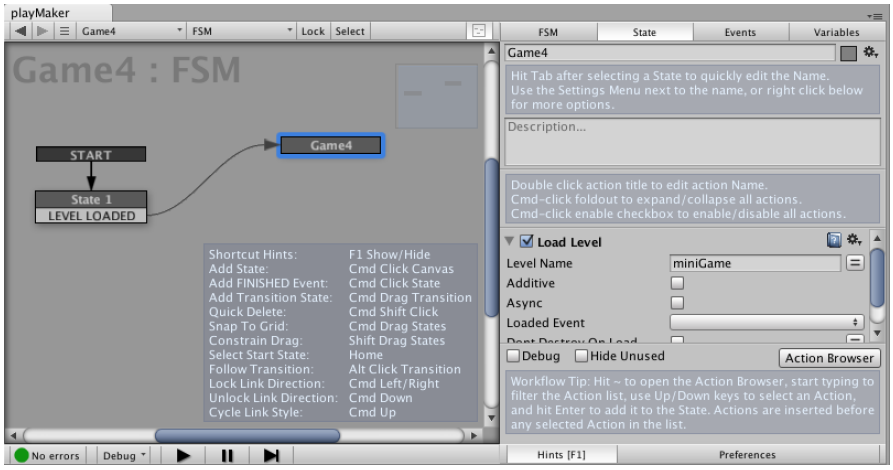
To use this menu in another game, you can just copy this scene to that game. If you want to add your game as a menu item in the existing game,

1. Open the SelectGame scene in Unity.
2. In the left tab, there will be Game1,Game2,Game3 under Canvas/Panel. Our objective is to add the new game as Game4 into it.
3. Duplicate Game3 in same place and rename it as Game4.

4. Click on Game4 and go to the playmaker tab. The states will be already defined there.
5. Rename state Game3 to Game4.
6. In FSM, select state ‘State1’. On the state tab to its right we can see it is a button.
7. Change the icon of the button with icon for your game and align the button top and left values.

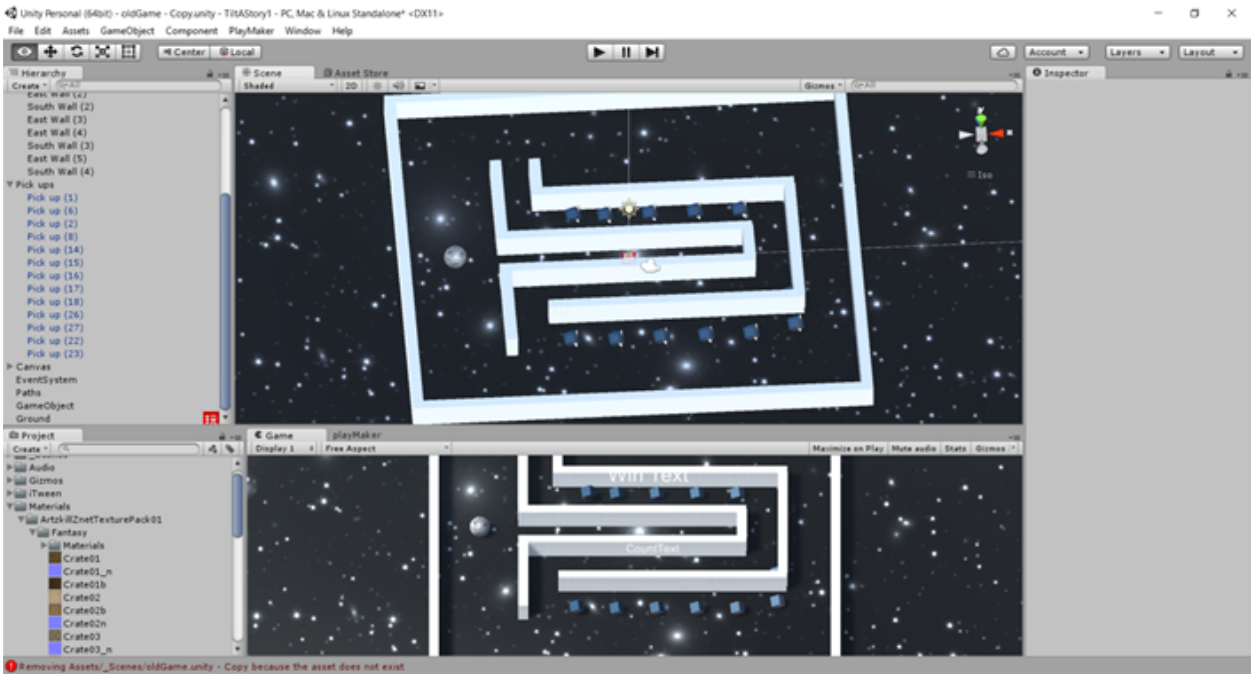


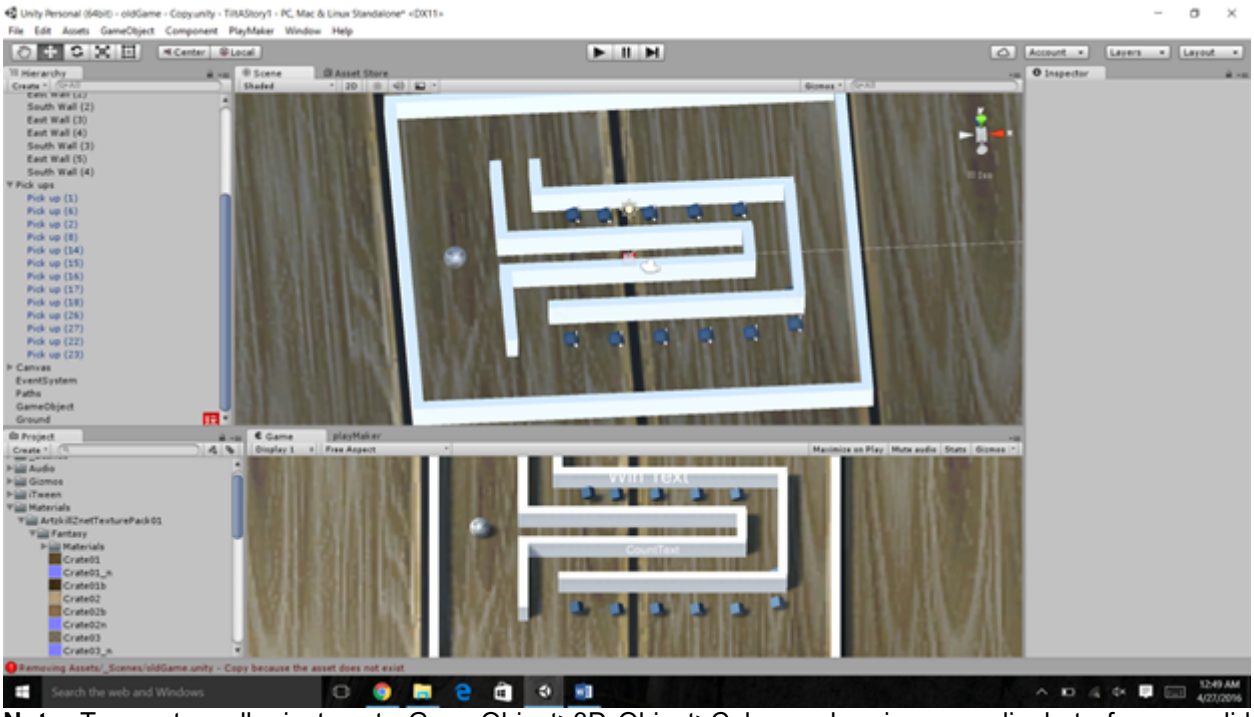
8. In FSM, select state ‘Game4’. On the state tab to its right change the ‘Level Name’ attribute value to the name of your scene. This scene should be present in list of build files. You can check this at File/Build settings.



1. Adding environment to the game :

You can add different backgrounds, colors and other texture to the ground as well as objects by just dragging and dropping the texture file onto the desired object.





Note: To create walls, just go to GameObject>3D Object>Cube, and resize accordingly to form a solid wall. To make the wall invisible in gameplay go to inspector window > uncheck mesh renderer property.

2. Adding characters and movement to the game:

You can add different objects as player or the pickups to the game.

To add a player object and get desired movement:

- Create a new GameObject
- Add player controller script in the inspector tab to the right.
- Place the player accordingly in the space.
- Player Controller script will handle the keyboard as well as accelerometer movements automatically.

To add a pickup:

- Add Player Controller Script.
- Add pickup object
- Change “TAG” in top right inspector value as Pick up
- In object Collider window: Check is Trigger property
- Add rigid body component from add component > physics > rigid body
- Check “Use Gravity” and “is Kinematic” checkboxes
- Run and Check if you are able to collect pickup or not.

Rotation:

- Select the game object to which you want to add rotation effect.
- In the inspector window, add component > add new script > Rotation.cs or Rotation1.cs

3. Adding Sounds to the game:

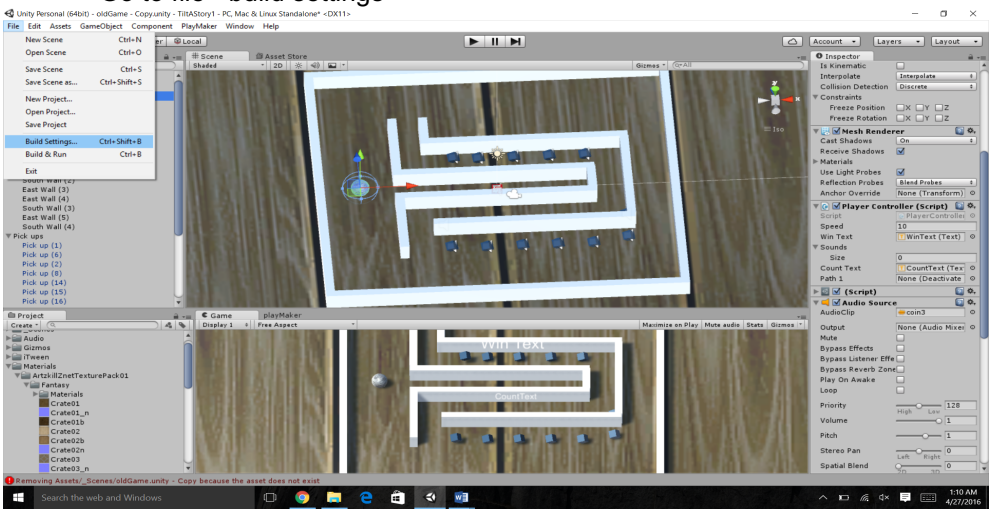
- Select the object > Add audio source component in inspector > browse audio clip property in

added audio source > add the desired sound.

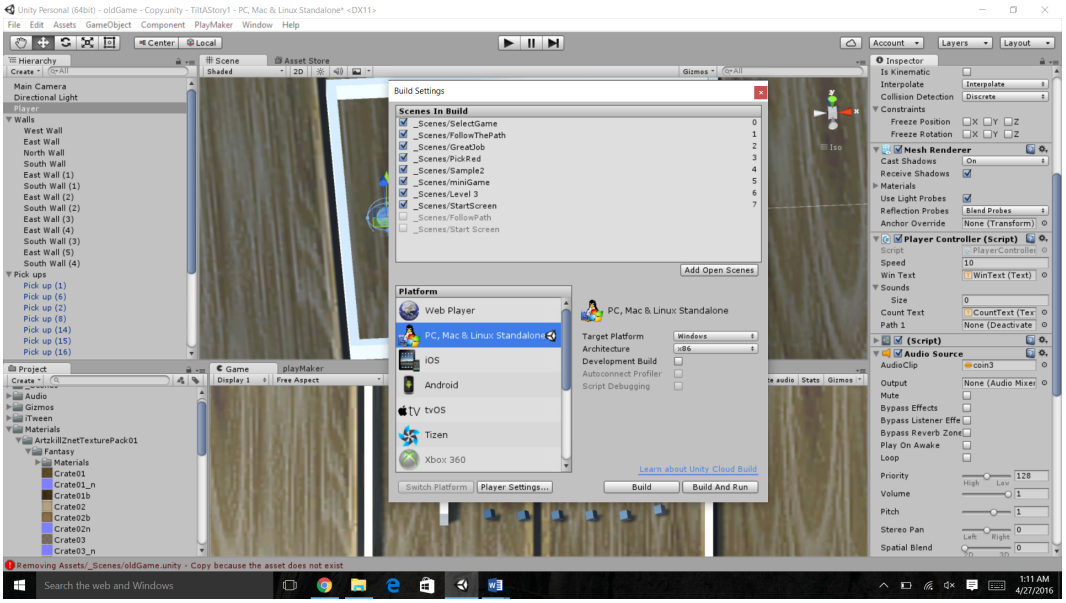
- If you want the sound at starting of scene, check “Play on Awake” property in audio source properties in inspector window.
- If it is a pickup object, uncheck “Play on Awake”

4. Compile and Building the game

- Go to file> build settings

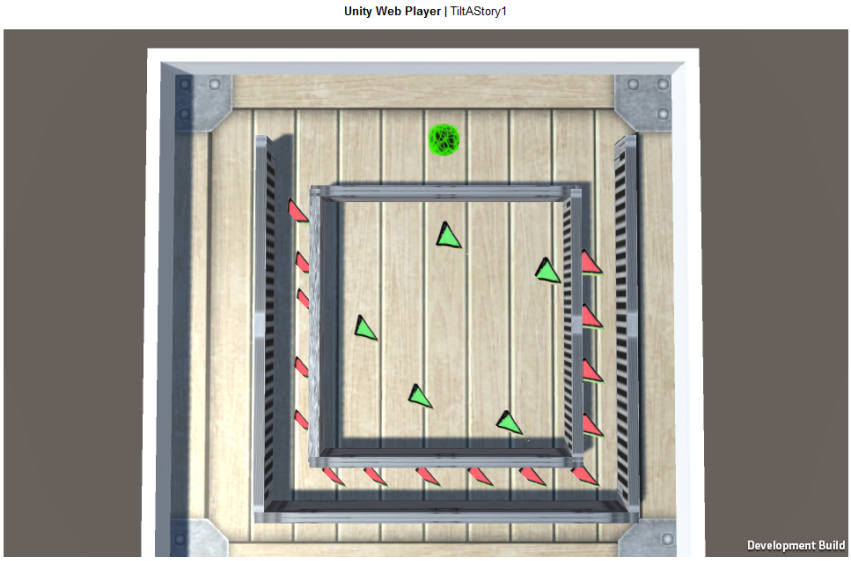


- Add and select all the scenes you want to build with first scene at the top and click “build” or “build and run”.



Designing a game using our Playmaker template:

We have developed a template which can be used to develop game levels without writing a single line of code and look like one showed below. This level can be loaded in Unity by opening ‘Level32d’ scene from Assets -> _Scenes -> Level32d.



In the above level, the red and orange triangles are pickups which can be picked up by the player using the green ball. When the player picks up all the red triangles, one of the walls is destroyed to gain access to green triangles. All of the movement of the ball can be controlled by using accelerometer of the mobile device.

Design:

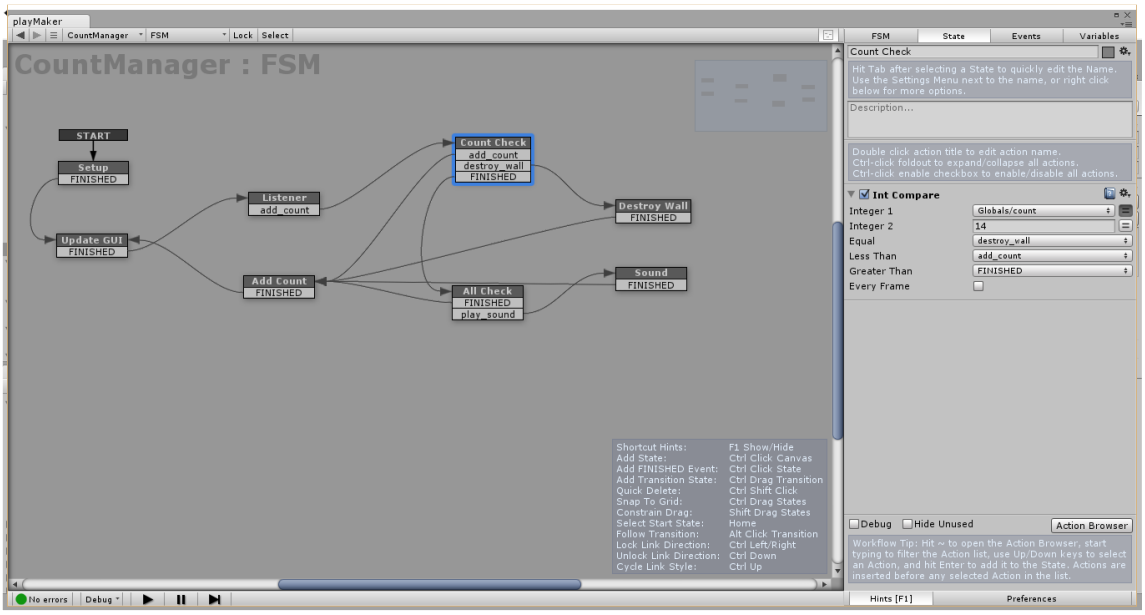
As shown above, the level and its corresponding hierarchy looks like this. The design of the environment and components such as paths, pickups, wall is up to the designer. The designer

can design the layouts with basic knowledge of Unity game engine. To create additional pickups, designer can simply right click one of the pickups and ‘duplicate’ it and change its position as desired.

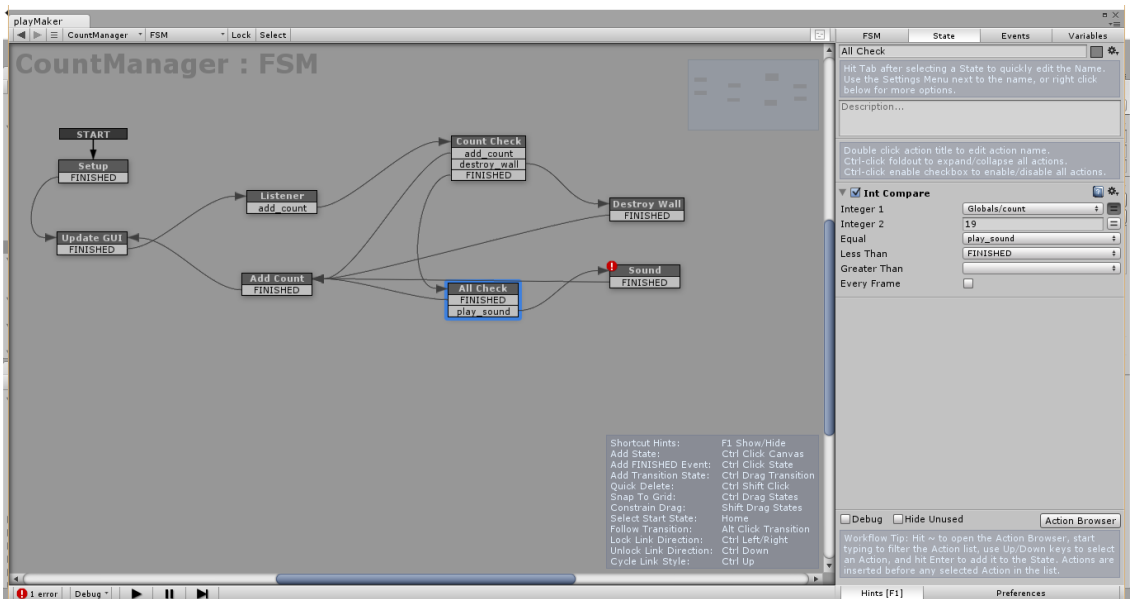
Gameplay:

Depending on the number of pickups, the designer can decide after how many pickups the wall should be destroyed. In our example, we have created 15 red pickups and 5 green pickups. Thus, we have designed it in a way that after all of the 15 red pickups are picked up, the wall will be destroyed. This number can change depending on the pickups the designer has added to the level and can be changed by following way:

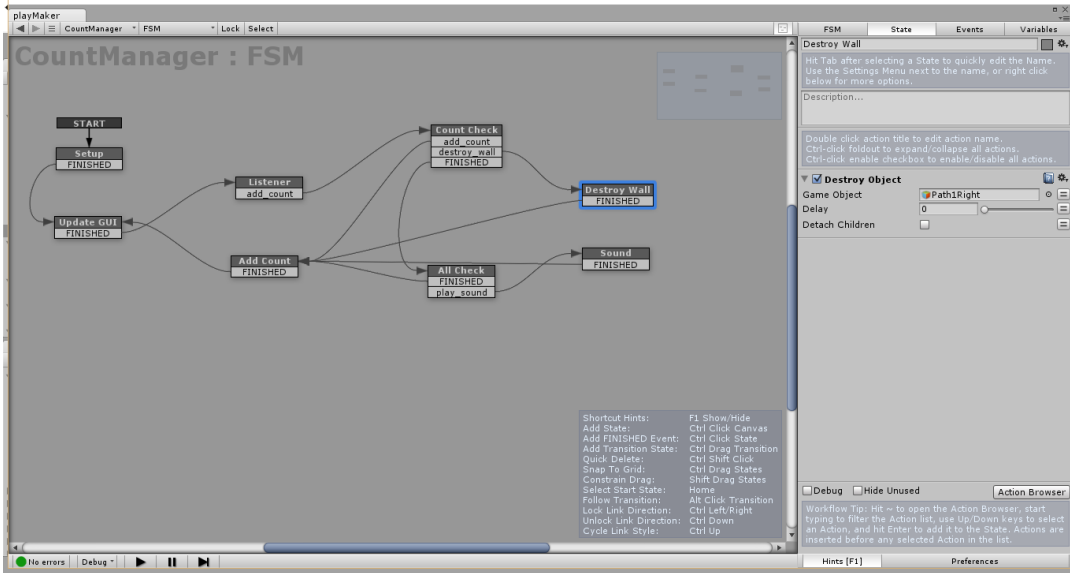
1. Open PlayMaker Editor.
2. Click on ‘CountManager’ and open its FSM.
3. Click on ‘Count Check’ state.
4. Edit the value of ‘Integer 2’ to number of pickups minus 1. As we had 15 pickups, we set it to 14.



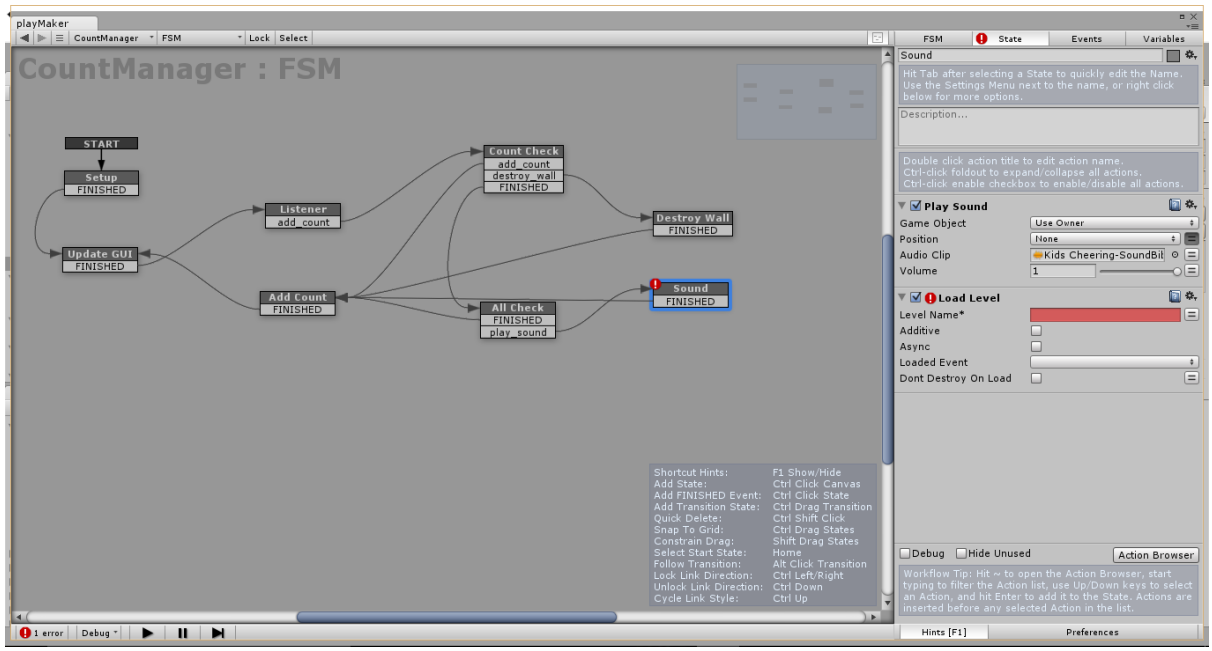
5. Similarly, value of ‘Integer 2’ in ‘All Check’ state can be edited in order to fire up an event that implies that the level has been completed. We had set the value to 19 as we had 20 pickups in total. The designer can set this value to number of pickups minus 1. This will fire up ‘Sound’ state.



6. To destroy a wall or any game object (multiple walls), go to ‘Destroy Wall’ state and select the ‘Game Object’ to be destroyed. In our case, we have selected it to be one of the walls. We are destroying just one wall but multiple walls can be destroyed by simply right clicking ‘Destroy Object’ action and selecting ‘Copy Selected Actions’ and ‘Paste Selected Actions’ in the same place.



6. To design desired actions after all the pickups have been picked up, go to ‘Sound State’. Here you can select the sound to be played by loading it in ‘Audio Clip’ variable. To do this, the audio clips will need to exist in ‘Audio’ folder of Project.
If you want to take the player to next level, this can be done through ‘Load Level’ action. Here the designer can simply mention the level that should be loaded at the end of current level.


playMaker
CountManager
FSM
Lock
Select
CountManager : FSM
START
Setup
FINISHED
Update GUI
FINISHED
Listener
add_count
Add Count
FINISHED
Count Check
add_count
destroy_wall
FINISHED
Destroy Wall
FINISHED
Sound
FINISHED
All Check
FINISHED
play_sound
Shortcut Hints: F1 Show/Hide
Add State: Ctrl Click Canvas
Add FINISHED Event: Ctrl Click State
Add Transition State: Ctrl Drag Transition
Quick Delete: Ctrl Shift Click
Snap To Grid: Ctrl Drag States
Constrain Drag: Shift Drag States
Select Start State: Home
Follow Transition: Alt Click Transition
Lock Link Direction: Ctrl Left/Right
Unlock Link Direction: Ctrl Down
Cycle Link Style: Ctrl Up
1 error
Debug
FSM
State
Events
Variables
Sound
Hit Tab after selecting a State to quickly edit the Name.
Use the Settings Menu next to the name, or right click below for more options.
Description...
Double click action title to edit action name.
Ctrl-click foldout to expand/collapse all actions.
Ctrl-click enable checkbox to enable/disable all actions.
Play Sound
Game Object
Use Owner
Position
None
Audio Clip
Kids Cheering-SoundBit
Volume
1
Load Level
Level Name*
Additive
Async
Loaded Event
Dont Destroy On Load
Debug
Hide Unused
Action Browser
Workflow Tip: Hit ~ to open the Action Browser, start typing to filter the Action list, use Up/Down keys to select an Action, and hit Enter to add it to the State. Actions are inserted before any selected Action in the list.
Hints [F1]
Preferences