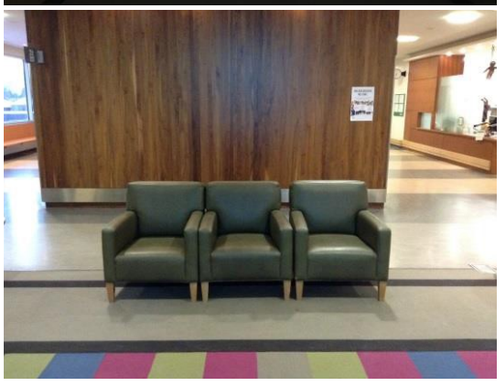
ScreenPlay

Environment Specifications

ScreenPlay is installed on the second floor of Holland Bloorview Kids Rehabilitation Hospital, in the out-patient clinic waiting space. To give you a better idea of the physical environment that ScreenPlay operates in, several pictures of the surrounding area are shown below.



ScreenPlay - Interactive display for kids at Holland Bloorview:

https://www.youtube.com/watch?v=wcEGkuRCQkA

More on ScreenPlay:

http://hollandbloorview.ca/newsarticle/introducing-ScreenPlay

The ScreenPlay Floor Pad consists of 100 pressure sensitive sensors (essentially on/off buttons) arranged in a 10x10 grid. Each tile measures 1 ft x1 ft. So, the entire flood pad measures 100 sq ft. The sensors are connected to a custom designed circuit board that is powered by an Arduino. The Arduino is connected to a Mac mini on which the visualization software runs.

Hardware Specifications

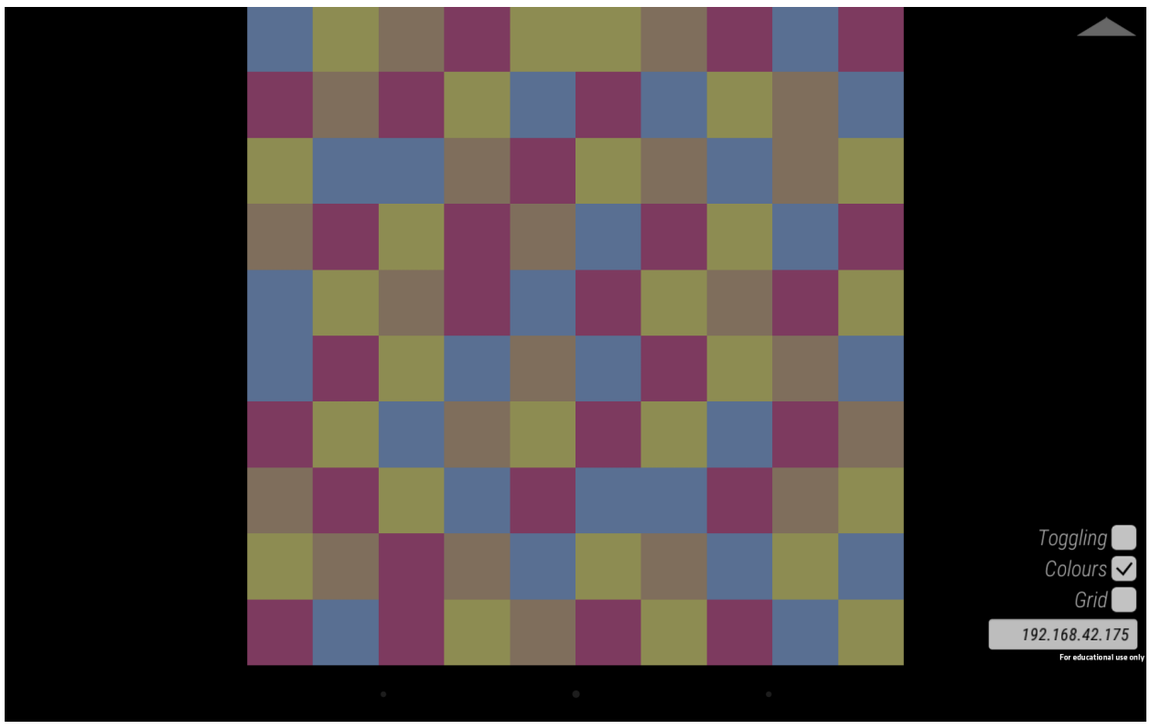
The Mac mini that will run the visualization software has the following specifications:

- Operating system installed is OS X 10.7.2
- Has a 2.7GHz dual-core Intel Core i7, with 8 GB of 1333 MHz DDR3 memory and an AMD Radeon HD 6630M graphics processor with 256MB of GDDR5 memory
- Display resolution is **1024 x 768**. Your visualization should be the same resolution as well
- The Mac mini has no access to the internet

The following projects have been tested with the current version of Unity 5.x (5.6.4f1). If you are using a version of Unity 2017.x, you may need to run Unity's built-in API updater when you first open each project.

ScreenPlayFloorSim

ScreenPlayFloorSim is an application that simulates the functionality of the ScreenPlay Floor Pad.



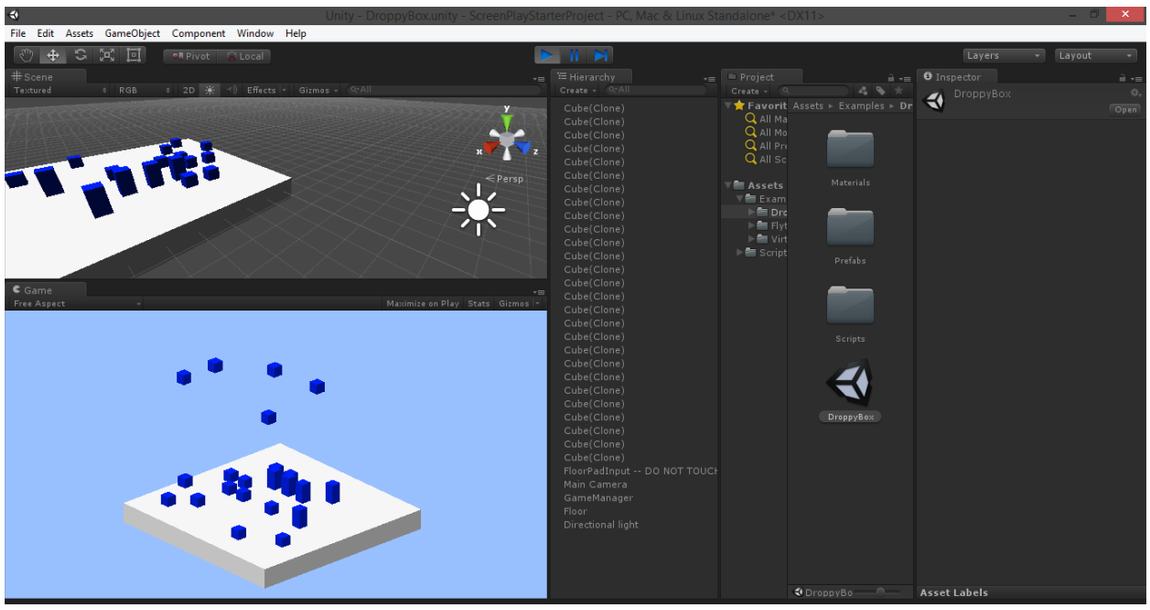
ScreenPlayFloorSim communicates via a network connection using the OSC protocol. Before launching the app, the simulator device and the development computer must be on the same network. Set the IP Address in ScreenPlayFloorSim:

1. Launch ScreenPlayFloorSim.
2. Tap the IP address Input Text Field.

3. Enter the IP address of the development computer running the ScreenPlay app you are building. **Note:** If you have built standalone versions of ScreenPlayFloorSim and your ScreenPlay project and are running both on the same development machine, you should use the IP address 127.0.0.1.

Test the Connectivity

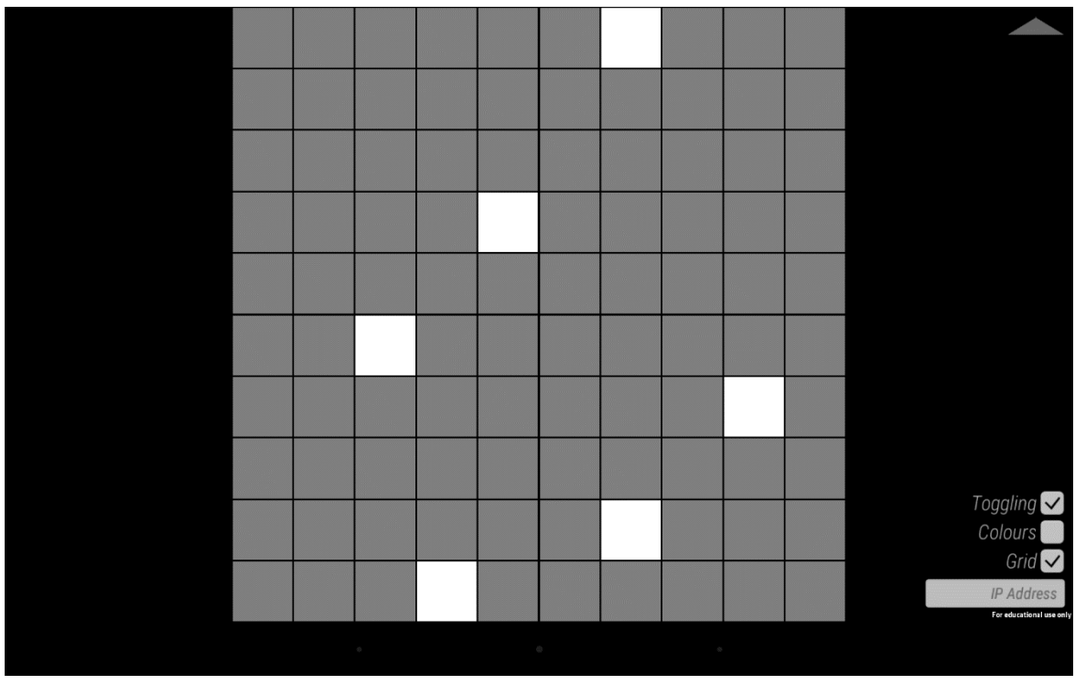
1. Open the **StarterProject** Unity project.
2. Open one of the Example Projects from the *Examples* subfolder in the Assets folder. (DroppyBox, Flythrough, VirtualFloorPad)
3. Press Play in Unity.
4. Press the tile buttons on ***ScreenPlayFloorSim*** app. You should see a response in the Example project.



ScreenPlayFloorSim Options

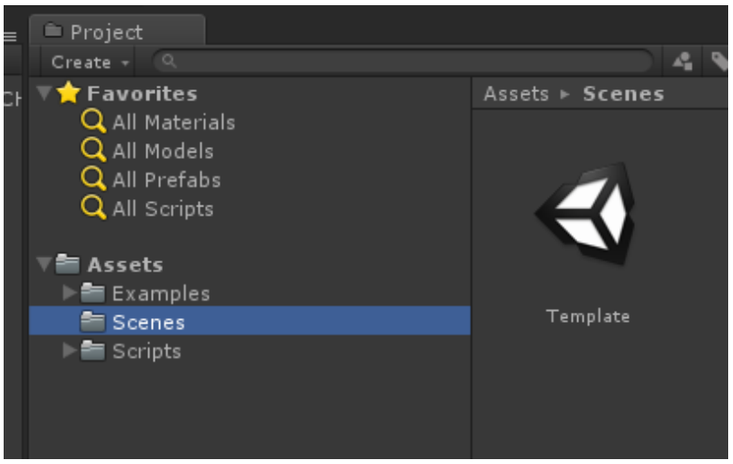
There are a few options available in ScreenPlayFloorSim.

- **Toggling**: When enabled, toggling changes the behaviour of the buttons in the app. This allows you to simulate several tiles being pressed at once without actually having to press them all at the same time.
- **Colours**: Allows switching between the replicated colour pattern of the actual floor pad and a grey-white setup. It is easier to see the tile state in the grey-white setup. Also, if the game you are developing does not make reference to the colour of the tiles, then this may be the preferred view.
- **Grid**: Adds grid lines between the tiles.

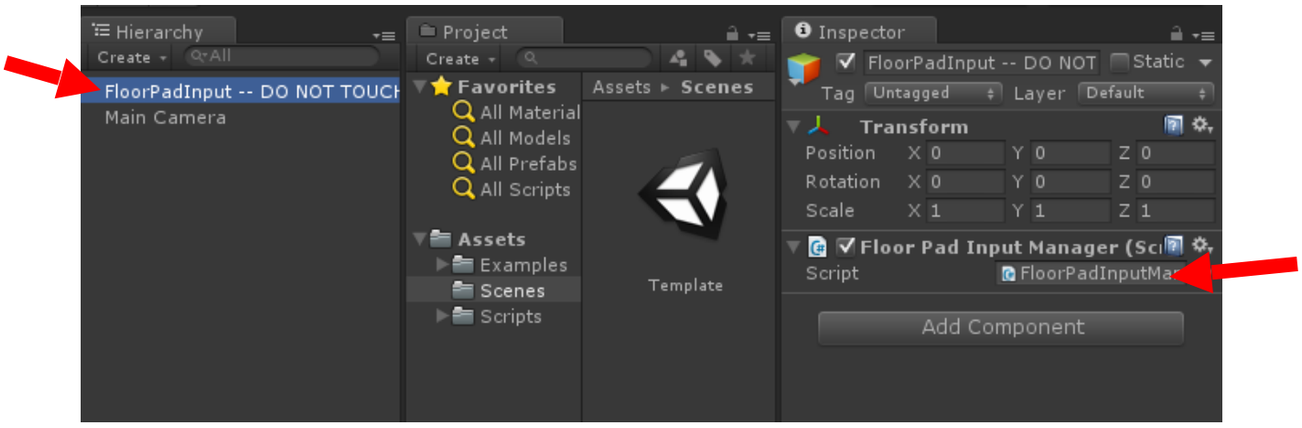


Unity Project Setup

1. Open the **StarterProject** Unity project
2. Open the Template scene in the Scenes subfolder.
3. This *nearly* empty scene is set up to accept the Floor Pad input. It is recommended that you save a copy of this scene for each scene in your game. File…Save Scene As… (Ctrl+Shift+S)



4. Take note that there is a GameObject in the Hierarchy named **"FloorPadInput – DO NOT TOUCH"**. This Game Object must be kept in each scene. This GameObject has a script attached to it that is responsible for receiving the ScreenPlay floor pad input.



FloorPadInput API

The FloorPadInput class has been written to resemble how keyboard input is handled within Unity.

FloorPadInput

Static Functions

`GetTile`	Returns true while the user holds down the tile identified by its coordinates.
`GetTileDown`	Returns true during the frame the user starts pressing down the tile identified by its coordinates.
`GetTileUp`	Returns true during the frame the user releases the tile identified by its coordinates.
`GetPressedCount`	Returns the number of tiles pressed. Guaranteed not to change throughout the frame.
`GetReleasedCount`	Returns the number of tiles not pressed. Guaranteed not to change throughout the frame.
`GetPressedCoordinates`	Returns a list of the tile coordinates of the tiles that are currently pressed.
`GetReleasedCoordinates`	Returns a list of the tile coordinates of the tiles that are currently released.

Events

The FloorPadInput class can provide event messages, similar to `OnCollisionEnter`, to your scripts.

In order to receive the FloorPadInput events, the following line must be added to your script's Update function:

```
FloorPadInput.GetEvents (gameObject)
```

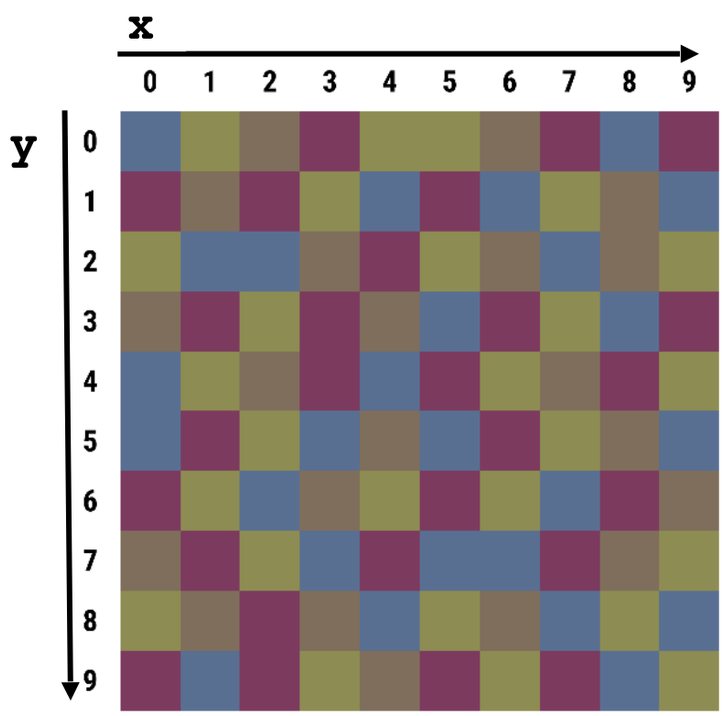
Example:

```
void Update ()

{
      FloorPadInput.GetEvents (gameObject);
}
```

`OnTilePressed (Vector2 coords)`	Returns the coordinates of the tile pressed during the frame. This function will be called as many times as tiles are pressed during the frame.
`OnTileReleased(Vector2 coords)`	Returns the coordinates of the tile released during the frame. This function will be called as many times as tiles are released during the frame.

Tiles are identified by their coordinates. (x, y)



Usage Examples

Look over the Example scenes provided in the Starter Project. There are examples on how to use the FloorPadInput **Events** there. Below are some simple examples.

GetTile

```
void Update ()

{
      if (FloorPadInput.GetTile (0, 0)) {
            Debug.Log ("The top left tile is pressed.");
}
}
```

GetTileDown

```
void Update ()

{
      if (FloorPadInput.GetTileDown (9, 9)) {
            Debug.Log ("The bottom right tile was pressed this
frame.");                  }
}
```

GetTileUp

```
void Update ()

{
      if (FloorPadInput.GetTileUp (0, 0)) {
            Debug.Log ("The top left tile was released this frame.");
}
}
```

OnTilePressed

```
void Update ()

{
      FloorPadInput.GetEvents (gameObject);
}

void OnTilePressed(Vector2 coords)

{

      if(coords.y > 4.5f){

            Debug.Log ("A bottom half tile was pressed this frame.");

      }

}
```