Campaign Editor Design

Overview

The goal of the Campaign Editor (CampEd) is to make levels easy to make. This editor will focus on providing story and action elements in the form of cutscenes, dialogue and events. To bring these together a Timeline is needed to organize and trigger events.

In general levels in the campaign / story will follow a simple structure of:

1. Comic Cutscene
2. Dialogue
3. Objectives
4. Dialogue
5. Boss
6. Dialogue
7. Comic Cutscene

Outline of Features for CampEd:
- Dialogue system
- Adding an objective system
  - Defeat Boss (or other enemies)
  - Plant bomb
  - Collect X amount of things
  - Survive for X amount of time

    - Kill X amount of enemies
- Event / Timeline
- Mini Scripting System
- Cutscenes (MSS)
- Spawning (MSS)
- Counting Stats
    - Time passed
    - Time from last event
    - Enemies on screen
    - Enemies killed
    - Health Remaining
    - Shots fired
    - etc….

Notes On Look and Feel:
- Separate Windows
- Window for Timeline, drag and drop
- Custom Editors for each object
- In SceneView gizmos
- Name (to identify tool)

Dialogue System

Overview

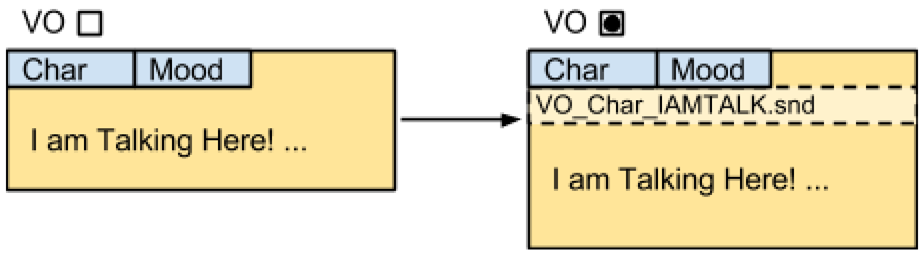
The dialogue system will be used to set up the interchanges between characters when they are in game. The System should support several features such as typing text, voiceover options, character images and moods, and standalone dialogue object.

Features

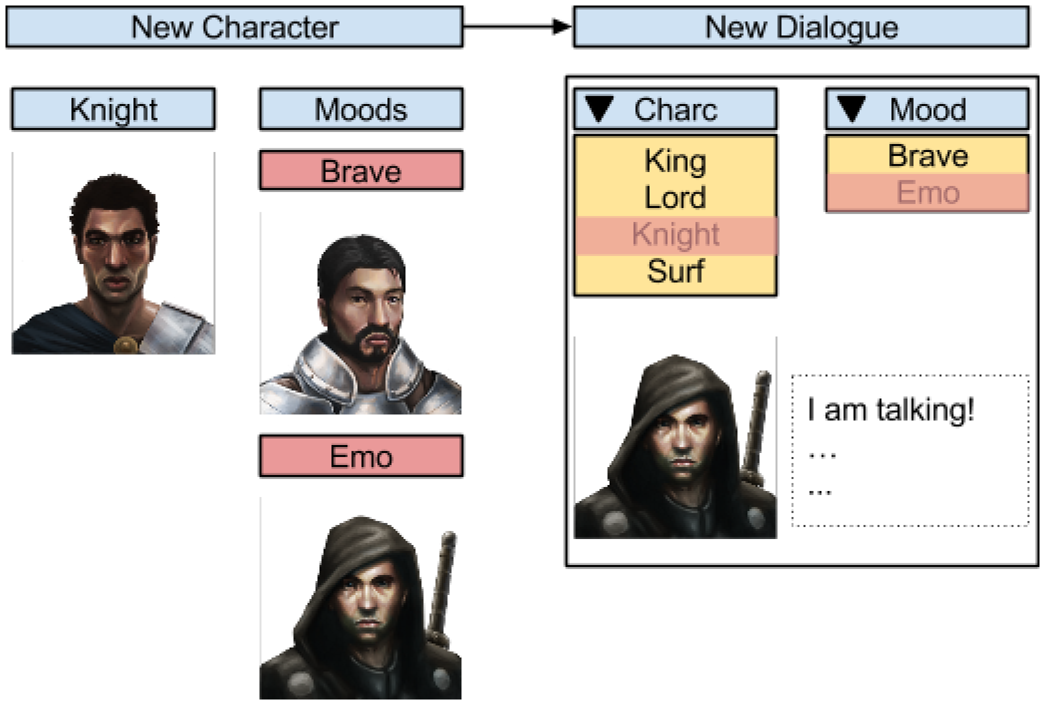
Typing Text - this is where the text is typed out on screen, either letter by letter or word by word. This is so that the players can read along as if the characters are speaking. Another aspect would be possibly sound bits. This is for when characters are not voiced and it adds much mood and life to the characters that are speaking.

Example of feature, **LINK**

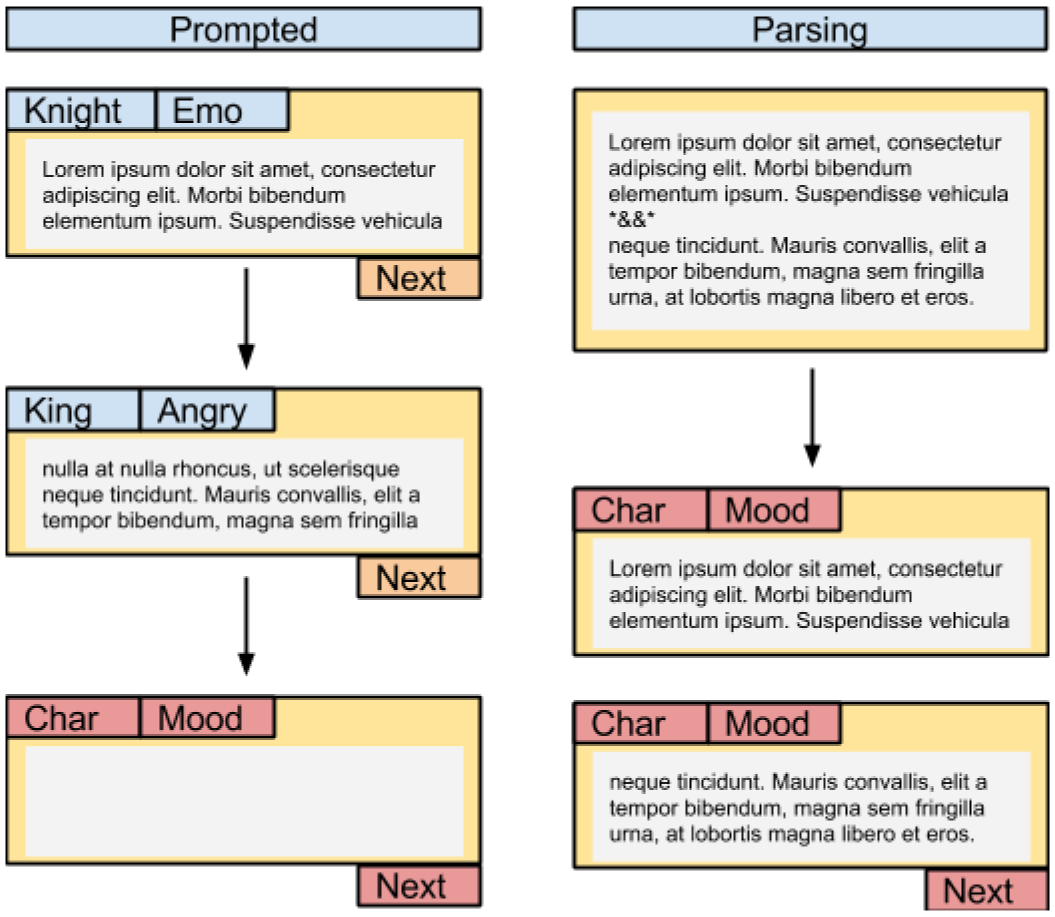
VO Option - An option when creating the dialogues should be to added a voiceover sound clip to the text. A simple check at the beginning allowing for audio files to be added to each portion of the text.



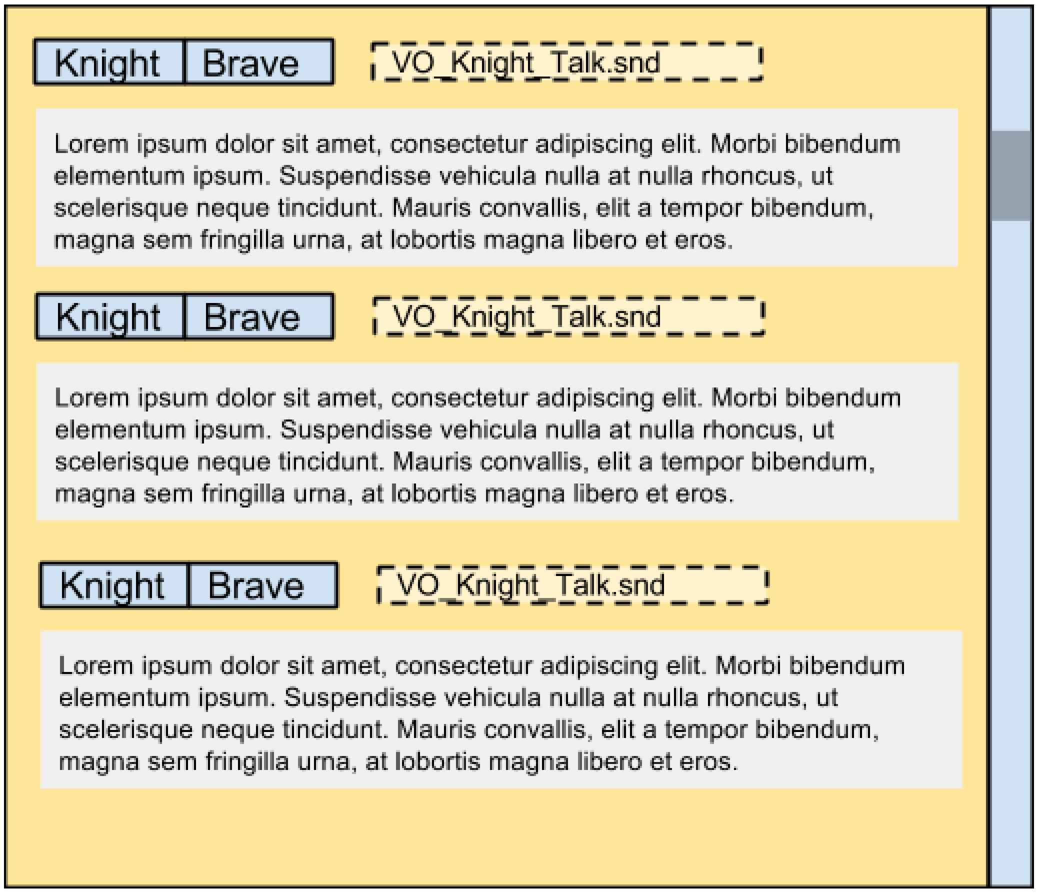
Characters - Each speaking character needs a portrait. This will then be displayed Alongside the text box. In addition to just the portrait different moods of the characters may be a part of the dialogue. In either case a Character should be able to be made before hand that stores it's name, it's image and images of it's mood. When entering dialogue the user just needs to select a character from the database and then it's mood.



Text Input - There needs to be two ways to enter text for the dialogue. For the first method the user is prompted with a text field to enter in the text for the current character. When that character's lines are done the user click's next and then enters in the liens for the next character. Simple and sequential. The second method is for text that has been written out already. Using the first method and copy/pasting is slow and error prone. Thus the second method would be using a parsing of the whole text. Using a unique symbol the text would be broken up into 'paragraphs' and then split when imported. Once imported the user would go through each one and assign a character to it.


Prompted
Knight
Emo
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi bibendum elementum ipsum. Suspendisse vehicula
Next
King
Angry
nulla at nulla rhoncus, ut scelerisque neque tincidunt. Mauris convallis, elit a tempor bibendum, magna sem fringilla
Next
Char
Mood
Next
Parsing
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi bibendum elementum ipsum. Suspendisse vehicula *&&* neque tincidunt. Mauris convallis, elit a tempor bibendum, magna sem fringilla urna, at lobortis magna libero et eros.
Char
Mood
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi bibendum elementum ipsum. Suspendisse vehicula
Char
Mood
neque tincidunt. Mauris convallis, elit a tempor bibendum, magna sem fringilla urna, at lobortis magna libero et eros.
Next

Saving and Editing - Once the dialogue is done it will be saved. For editing purposes it is easier to be able to see all dialogue with all the attributes at once. The format in which to display the information is to be like that of a word processing document. See image below. From this view the user would be able to edit all aspects of the dialogue, lines, character, mood, etc.


Knight
Brave
VO_Knight_Talk.snd
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi bibendum elementum ipsum. Suspendisse vehicula nulla at nulla rhoncus, ut scelerisque neque tincidunt. Mauris convallis, elit a tempor bibendum, magna sem fringilla urna, at lobortis magna libero et eros.
Knight
Brave
VO_Knight_Talk.snd
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi bibendum elementum ipsum. Suspendisse vehicula nulla at nulla rhoncus, ut scelerisque neque tincidunt. Mauris convallis, elit a tempor bibendum, magna sem fringilla urna, at lobortis magna libero et eros.
Knight
Brave
VO_Knight_Talk.snd
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi bibendum elementum ipsum. Suspendisse vehicula nulla at nulla rhoncus, ut scelerisque neque tincidunt. Mauris convallis, elit a tempor bibendum, magna sem fringilla urna, at lobortis magna libero et eros.

Stand-Alone Object - Once made the dialogue should be accessible from a single object that does not need other objects or code to work and once activated in a scene will play out it's dialogue.

Other Features

Display

Setting Font - Should be able to set different font for different characters and set color, style and size. (If possible also to be able to set the style and color inline in the text).

Position - Setting the location of the text box in the screen, left or right, top or bottom.

Work Flow - The flow of the system should be:

1. Written Dialogue by Writer.
2. Designer Creates Characters in System.
3. Dialogue is assigned to Characters.
4. Assign Specials to Dialogue.
5. Saved in format that can be edited.
6. Saved as an prefab to be instantiated.

Objective System

Overview

The Objective System is a simple tool that allows for predefined objective types to be placed in a level and then the attributes changed to suit the level and story. These objectives are then registered with the Timeline so they active in order in the level.

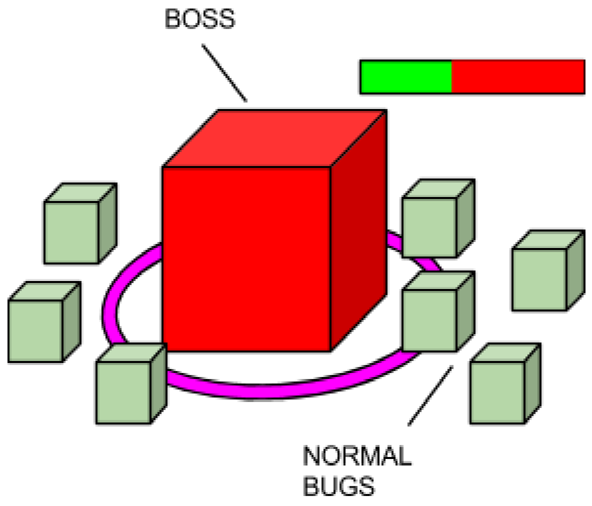
One aspect of this feature is to decide if all objectives should be stored in a single 'Controller' object, so each objective is added via the inspector for the 'Controller' in a custom editor, or if each objective should be their own objects.

Gizmos

Many of the objectives need to be set up in the Scene View when editing the game. To aid in this most of the objects and points used by the objectives need a custom gizmo to show that they are a part of an objective or mission. Other Gizmos (and Handles) will be needed to help place, edit and debug the objectives. An example is points, instead of using empty game objects a Handle is made to move a coded point the game space, or using gizmo lines to show relations, direction and distance, when the game is playing and the objective is active.

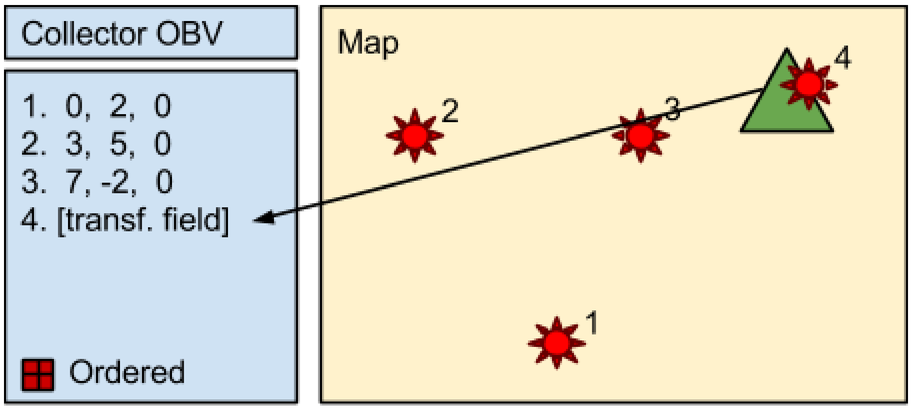
Types of Objectives

Defeat Enemy (Boss) - This objective is simple with the aim being to destroy the targeted enemy. When making this objective the enemy from the scene is placed in into the objective. When the game plays an indicator ring should be displayed around the enemy and a health bar as well (since these are important enemies with high health).



Courier - The Courier objectives are ones where the players have to go to a certain to drop off point. This is a simple objective, there is no prerequisite for going to the drop off point. An option to have an ingame compass arrow, this arrow would point to the drop off point. A custom gizmo is needed, also for debugging line gizmos to show how far the player is from the goal (maybe have this also displayed in Play Mode as well as the editor Scene View).

Collector - The Collector objective is where the players have to collect a certain object or objects. While similar to the Courier the Collector objective has to support multiple locations / objects and keep a check list of collected items, and as an option make sure objects are picked up in the correct order (not necessary for all of them). An optional compass arrow which shows the direction of the next pick up point if ordered, or all pickup points at once (ie. an arrow displayed for each point)   The editor UI has to be able to add pick up points. In Scene View there should be gizmos showing the locations and be able to move the points around the map. Another option is to drag and drop an object into the editor so that that objective point is tracked (this will be useful for pre-placed objects and objects that might move). Again a custom gizmo logo to identify the points in Scene View and counters if there is an order for the object to be picked up in.



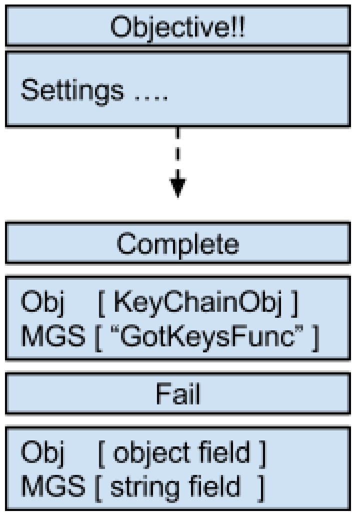
Survive - Survive is a simple objective where the players have to stay all alive while the timer is running. While the objective is running a timer should show with the remaining time left. The editor is simple with being able to set the length of time it should run.

Kill All Enemies - Kill All Enemies is a simple objective. The objective is set to killing a number of enemies. As each enemy is killed a counter goes up. Once the count has reached the set value the objective is complete. When running in game a counter needs to be displayed to show the number of kills made.

Features

Objective Notifications - When an objective is activated a notification needs to be played to the players. This contains the name and stats of the objective. Another notification is needed also when the objective is completed letting the player know that they finished (or failed). The fields for setting these can be simple string fields where a designer will type in the information (though stats should be added on via code so there are no differences between what is written and what is the actual objective).

End of Objectives - When an objective ends there are two outcomes. Either the player met the requirements and completed it, or the player failed. For each outcome a way to trigger the next event should avaliable. For core mission objectives it would move the timeline along activating the next event, and if not completed fail the level. For optional missions and objectives, a custom trigger for an event for both complete and failed, and if this field is left blank nothing happens. The way the trigger is done can be changed, but for a first go two fields should be available, one to type in a string which would then broadcast a message along the object, and a second field which a gameobject could be placed. If one is then the message is sent to that object.



Timeline and Events

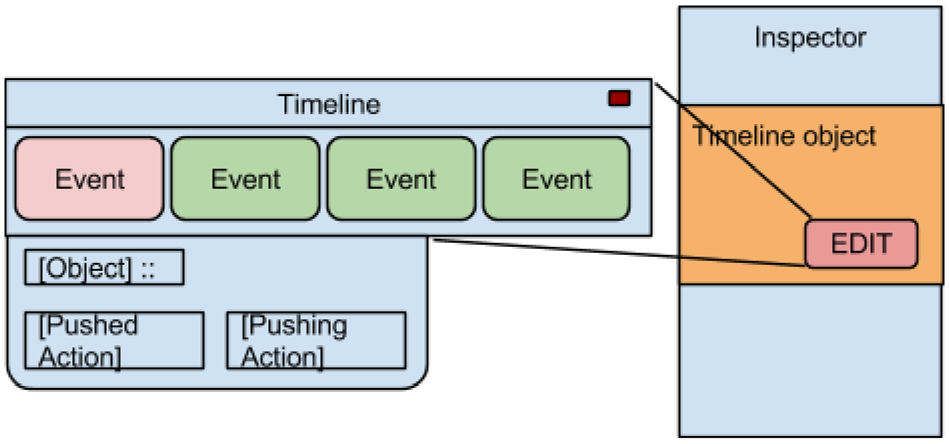
Overview

The Timeline System is a tool to help with sequencing of events in game. Most of the time events will happen in a chronological order. With in the general campaign dialogue and objectives will be the main events, as such certain dialogue only appears after an event has happened, or an objective will end pushing the next event to happen. Thus the timeline works to activate and deactivate objects that control the events.

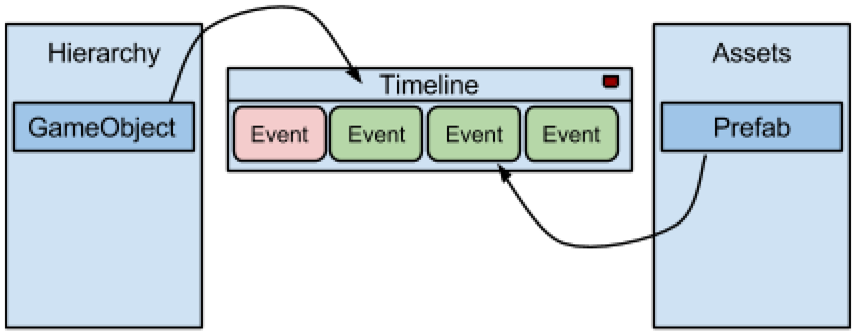
With the editor a timeline object will be created and placed in a scene. When editing the timeline the user can drag and drop new objects into the timeline window and rearrange events. Each event can have beginning and ending effects to customize and vary the use of objects with less coding.

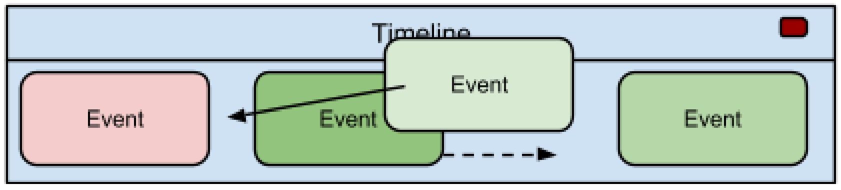
Features

Separate PopUp Window - When editing the Timeline object a custom UI in a separate window. This is the window in which objects can be registered as events, rearrange events, and edit the events.



Drag and Drop - When adding a new object as an event to the timeline a simple action of dragging and dropping gameobjects and prefabs into the timeline window will create the new event and add the appropriate scripts to the object if needed and registering the object, it’s event, and any action when the event gets pushed or the event pushes. Also dragging within the window to a new position should move the selected event to the new position in the timeline.

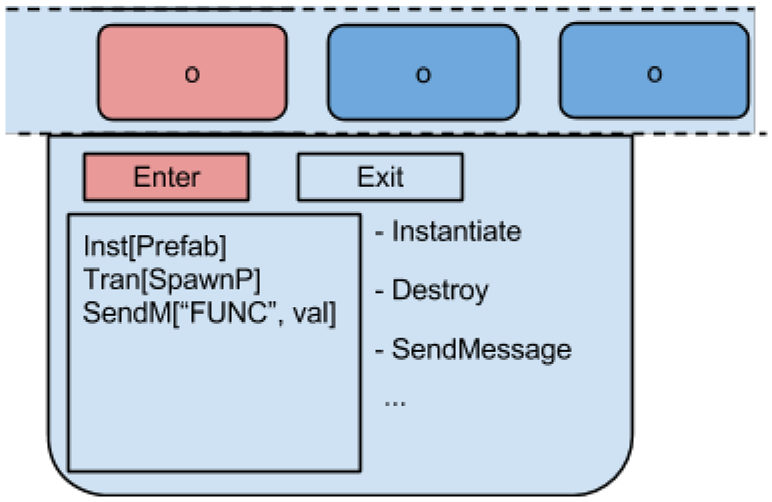




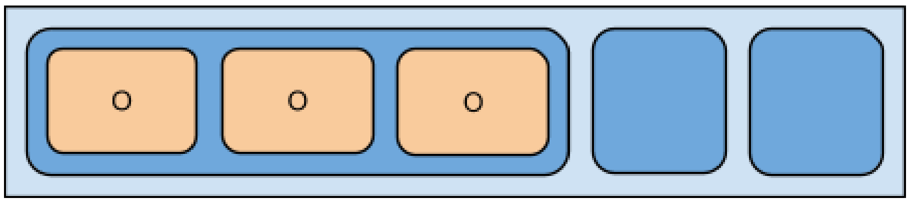
Options - For each event there should be an enter and exit action. When an event ends it pushes the timeline to the next event which activates the exit action and then the enter action to the next event. For these actions there should be a set of generic options that manipulate gameobjects and send messages, and a combination/multiple of them, ie. need to instantiate, transform, and send message. For ease default should be instantiate and destory for enter and exit.

A list of options are:

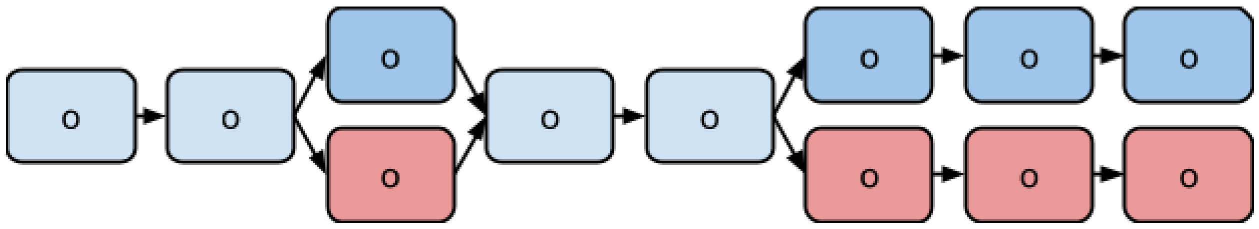
- Instantiate
- Destroy
- Activate/Deactivate
- Send Message
- Transform



Nested Events - At times more complex scenarios might need to be made. One way of making more interesting and dynamic levels is by nesting events inside other broader events. An example of nested events is where three small objectives are nesting in an objective to survive for 5 minutes. While the 5 minutes are still running the three objectives will still run as normal. However once the 5 five minutes are up, hence the event is done, the timeline will pass all the objectives that are nested and go on to the next event after the encompassing event.



Branching - Another way to make more complex scenarios is by branching events. When an event ends a fail or pass condition can be set so that the path the timeline takes changes. This can be used to make a whole alternate set of events for a whole level or just a single set of different outcomes.



User Workflow:

1. Create new Timeline for scene
2. Open Edit window for Timeline
3. Drag objects into timeline to create events
4. Rearrange events (make nests and branches too)
5. Edit Enter and Exit actions

Mini Scripting System

If time allows this will be developed. Until the previous tools are designed the MSS will be left as an outline.

Outline

1. A side addition to Objectives and Timeline.
2. Be able to type in code from unity editor.
3. Adds simple custom functionality to objects
    a. check variables
    b. run delegates
    c. basic functions (add components, find objects, create random number)
    d. do basic logic (check for collisions, sums)