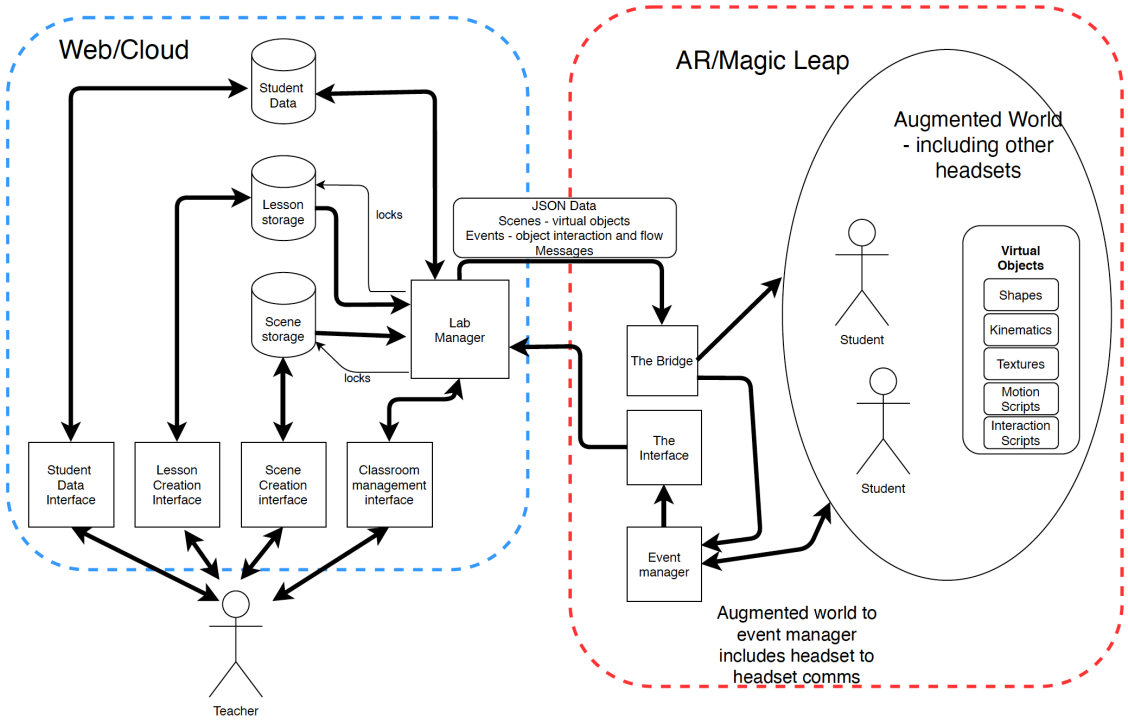
CyberAR System Design

Software overview

This is a rough design of the software needed for our CyberAR project. It is not in final form but rather designed to solidify the concepts we might use in this implementation.



Game Objects

First, we need to discuss the requirements for the virtual objects that live in our augmented

world. They have certain required and optional characteristics. They would include location, size, texture, shape, orientation, and the scripted behavior associated with the objects.

Let's address each of these separately:

- ## Object attributes

  - ### Shape

    - Objects will be assigned a primitive shape or be associated with a loadable shapefile. This shapefile will exist either internally in the ML unit in the assets folder or be loadable from the web from a specified URL.
    - If a shape is not specified, the object's shape will be a primitive sphere.

  - ### Size

    - The objects xs, ys, zs scale-size, will be set by the script.
    - If an object is not given a specified size, it will be given a unit size.

  - ### Location

    The script will set-The objects x, y, and z location.
    - The default location will be at (0,0,1) - 1 meter above the scene's center.

  - ### Orientation

    - An initial orientation of the object will be specified using Euler angles.
    - The default orientation will be (0,0,0)

  - ### Texture/Color

    The script will specify-The object texture in one of three ways:
    - The object will use a simple color as specified by an RGB triplet

- The object will use a named texture, which is preloaded in the unit
- The object will use a texture loadable from the web via a URL.
- Details on importing texture in Unity are here:
- https://forum.unity.com/threads/importing-image-as-texture-at-runtime-from-device.419000/

        - The default texture for an object will be a simple red color.

    - **Illumination**

        - Objects may give off light. If they do, the light will be spherically distributed. The user will specify a:
            - Intensity level
            - Cutoff range
            - If the scene lighting should be turned off.
        - Objects will reflect ambient light by default.

    - **Visiblility**

        - Objects may be flagged as invisible.
        - By default, objects will be visible.

- ## Object methods

    - The behavior of objects will be associated with scripts. These scripts will be in two categories.
        - Behaviors
        - Triggers/interactions

    - **Behaviors**

      Behaviors are the scripts that change the location, appearance, orientation, existence or other objects' observable characteristics.
        - Behaviors include:

        - Destructors - deletes an object
        - Rescaling - allowing an object to change its size
        - Relocation - moving an object
        - Motion - animating the motion of an object along a path at a given speed
        - Rotation - changing the orientation of an object
        - Rotators - animating the rotation of an object at a given speed
        - Textures - changing the texture of an object
        - Toggles
          -Visibility toggles - setting an object as visible or invisible
          -Interaction toggles - setting an object to have kinematic behavior or interactions
    - Available Behaviors must register with the event manager when the object is created.

- **Triggers**
    - Triggers are scripts associated with reporting particular events back to an events manager. They are not associated with internal behaviors.
    - Triggers might include:
        - A generic collision with another virtual object
        - A collision with a specific object
        - A timer started upon a particular event
    - Triggers can be configured at runtime like scripts.
    - The available triggers must register with the event manager when the object is created.

Managing the AR world on the device

There are three elements in the AR system that need to be created.

IO with the web system

The Bridge

- Translates information from the lab manager
- Grabs virtual scenes from the lab manager
  - Parses JSON and turns them into virtual objects
  - Grabs textures and objects as needed, based on the JSON
  - Connects objects to the event manager through bidirectional listening to interfaces
- Grabs Lessons to configure the behavior of the scene
  - Lessons are configurations of the event manager
  - The events such as touching an object, answer a question, motion, time, collisions, etc. trigger new scenes or behaviors
- Listens for messages form the lab manager, including
  - Flags to the bridge to update the scene
  - Other external inputs
- The bridge will have some standard interface elements pre-defined, including:
  - Standard pushbuttons
  - Sliders
  - Simple graphs - to do y data vs. x data plots
  - Other UI elements
- The JSON in the bridge will have inheritance either directly or through the scene writing interface.
  -The system should allow us to create a generic class of “planets” and then make a specific version of each planet.

The Interface

- The system that sends information from the device back to the lab manager
- Data includes:
  - Data to request a new scene or update
  - Student data such as logins, progress, or responses to questions
  - Requests from the device to update or refresh the scene.
  - Requests from the device to grab a new texture or object.
  - Events from the event manager as scripted.
  - Student status data - including 3d positions, orientations, etc.

Startup and shutdown

Startup

Several actions need to execute when the device is being initialized. These include:

- Logging the user into the system via an interface
- Determining the class, location, student, and instructor
- Scanning the environment to form a map of the area
- Associate this map with the coordinate system in the device
- Looking for other devices to join into lab groups
- Determining who serves the transmission objects

Shutdown

Upon shutdown, the system needs to do some cleanup procedures.

- If it is a member of a group, it needs to notify the device that is the server that it is disconnecting
- If it is the server of a group, it needs to hand this off to another device in the group.

        - The final state needs to be determined and logged into the webserver.

The Event Manager

- This event manager processes the events from the virtual world within each device and between devices. It also processes inputs between the virtual world and the lab manager via message through the interface. A series of events associated with a scene will be called a “lesson” since we are tying these interactions to pedagogical goals.
  This website talks about the scripting required for our system:
    - *https://gameprogrammingpatterns.com/bytecode.html*
- This event manager must be scriptable to allow virtual objects to register dynamically as they are added.
- Objects must register with the event manager as they are created to establish the bi-directional link.
- Objects must have a port open to receive events from the manager.
- Once an object is registered, events to and from it will become scriptable.
- Scriptable events could include:
    - An object being touched
    - A controller event
    - A hand gesture
    - A collision with another object
    - An object’s position is changed, or the object enters into a selected area
    - A flag from the lab manager indicated a change of state in the scene
    - An event from another ML device being transmitted to this device
    - Logical combinations of multiple events - for example, colliding with another object in a selected area
- Scriptable reactions could include:
    - Deleting an object or loading an object - or scene
    - Changing or loading a texture

- Saving data to a student record
- Changing the property of an object such as position, velocity
- Changing the appears of an object such as size, texture, or color
- Triggering an event to be transmitted to other devices
- Sending a flag to the lab manager
- A time delay - followed by an event
- Multiple reactions to the same event or events

Web/Cloud Lab Manager

The core element of the cloud design is the lab manager. This software mediates the interaction between the stored sciences, scripted interactions, and the databases storing them. It also allows a teacher to push live updates during the lab and monitor student progress in real-time. Any logging of student interactions and behavior happens through this interface.

The lab manager has two core portions:

- ## The Output Module
    - The output module for pushing data to the ML through the bridge. The data will include:
        - JSON files define objects, attached scripts, textures, shapes, location, etc. This is currently the input to the Bridge as described above.
        - Texture files that are not loaded on the ML device.
        - Shapefiles not on the ML device.
        - Other auxiliary files such as audio and video clips.
        - Messages from the lab manager to the devices. The message portal will be polled regularly or be pushed to an open port on the ML device.
        - The messages could include:
            - Instructions to update the current scene

        - Instructions to push the current status of the user to the database
        - Or any command that can be processed by the event manager.
    - Updates to the messages will be pulled from the database at regular intervals.

- **The Input Module**
    - The input module is responsible for receiving data from the ML.
        - Data includes:
        - Data to request a new scene or update
        - Student data such as logins, progress, or responses to questions
        - Requests from the device to update or refresh the scene.
        - Requests from the device to grab a new texture or object.
        - Events from the event manager as scripted.
        - Student status data - including 3d positions, orientations, etc.

Short term software goals

On the AR device:

- Implement all elements of game objects, including:
    - Generic attributes
    - Behavior scripts for changing the attributes
    - Trigger scripts that report events to the event manager
    - JSON parsing so all objects can be dynamically instantiated
    - Create bidirectional links to the event manager
- Implement a scriptable event manager
    - Tie game object creation to this script
    - Create a simple scriptable language to define the connections, including multiple

events and multiple actions
    - Create a standard JSON interface to define these interactions
- Extend the bridge to process textures (and perhaps OBJ files)
    - Update the bridge to include standard scripts and connections to the event manager
    - Update the bridge to receive information as events to the event manager
- Create the interface to post information to the website
    - Posting login information
    - Requests for new scene data
    - User information

On the website

- Create a web management system for the project
    - Sending data
        - It must be able to serve JSON objects that define a scene
        - It must be able to serve JSON objects that configure the event manager
        - It must be able to serve textures and perhaps OBJ files
    - Receiving data
        - It should be able to process requests for scenes
        - It should respond to routine queries for status updates as requested by the event manager through the interface
        - It should record user data