INTRO TO VWF SANDBOX TUTORIAL

OVERVIEW:

At GEC22, ADL and LMMST will present our flagship Virtual World Framework application: the Mars Game. The Mars Game subtly introduces programming and math content into a fun and rewarding 3D gaming experience. Students solve puzzles in a low risk environment as they learn to navigate a damaged lander across the surface of the red planet. In a study of 140 9th and 10th grade students, students learning with the Mars Game show a significant improvement in willingness to attempt programming challenges in a post intervention assessment. This session will explore the technology used to create the Mars Game.

The Virtual World Framework Sandbox is a web-based, HTML5, collaborative game and simulation design toolkit. It runs completely in the web browser, and is synchronized in real time between geographically dispersed users through a messaging queue on the GENI servers. In today's session, we're going to learn a bit about how to create a game experience similar to the educational experience discussed in (insert name of Mars Game Session). We'll also take a moment to review the ADL Mars Game project. Participants will:

1. Import art assets from a provided library, arrange them in the scene, and set basic properties like the object name and location.
2. Add scripts to objects, edit those scripts, and manipulate object properties through JavaScript
3. Setup the physics engine to receive notifications about object intersections, and handle those notifications in code
4. Play, test, and publish the finished experience.

In pursuit of these goals, it is expected that the participants will learn the basics of the Sandbox user interface, the general structure of the engine and website, and the principals behind the VWF architecture.

PREREQUISITES:

- Basic understanding of **JAVASCRIPT** syntax is recommended, but not required.
- A familiarity with basics web programming topics like HTML, and CSS
- Previous experience in other 3D immersive environments will be helpful, but is not required
- The participants should possess a computer which
    - Includes an **UP-TO-DATE** web browser (Chrome, Firefox or Safari)
    - Has hardware accelerated 3D capabilities (anything with a dedicated GPU will do)
    - Supports **WEBGL** and **WEBSOCKETS**, and is connected to the conference network

RESOURCES:

- The VWF Sandbox GitHub site (http://github.com/adlnet/Sandbox)
- The Sandbox WIKI (https://github.com/adlnet/sandbox/wiki)
- Does your computer support WebGL? (http://doesmybrowsersupportwebgl.com/)

TUTORIAL INSTRUCTIONS:

1. Test your web browser for **WEBGL** support
2. Sign up for an account on the Sandbox server
3. Allocate a new world in which to work
4. Read over the background reference and learn the camera motions

1. Setup the environment with objects from the content library
2. Assign the scripts to the objects
3. Build the physics collision proxy objects
4. Create the failure and success screens
5. Write the code to hook up the failure and success conditions

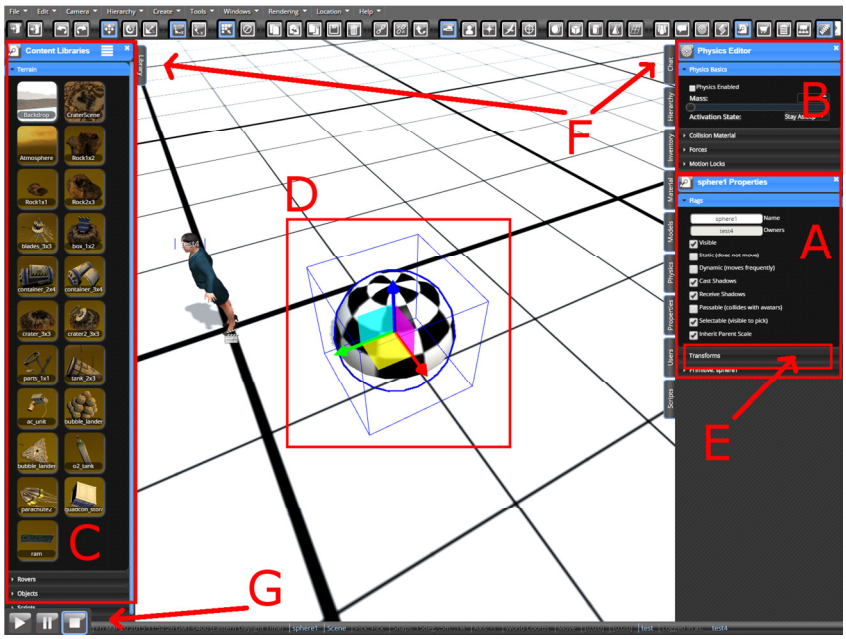
1. Test gameplay while in the editor
2. Publish out your game
3. Share and review.
4. Delete your game (optional)

BACKGROUND AND REFERENCE:

MOVING AROUND SPACE

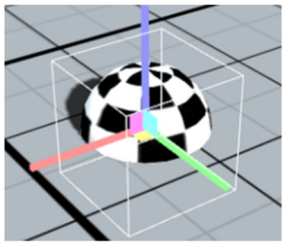
The ability to move around a virtual environment is critical. When in Editor Mode, the GUI provides many ways to manipulate the active camera view. For the majority of this tutorial, you'll be in 'Orbit' mode. This mode orbits the camera around a selected point while always facing toward the center. When in orbit mode:

- If you have a 3-button mouse
  - **RIGHT-CLICK** and drag the mouse to left or right to rotate around the selected object or point
  - **RIGHT-CLICK** and drag up or down to move the camera above or below the selected point
  - Push and hold down the **CENTER MOUSE WHEEL OR BUTTON** to pan the center point

    - Roll the **MOUSE WHEEL** to zoom toward or away from the selected point
    - Click **'FOCUS TO SELECTED OBJECT'** on the toolbar to move the center to the selected object
- If you have a keyboard
    - Press and hold the **LEFT OR RIGHT ARROWS** to rotate around the selected object or point
    - Press and hold the **UP OR DOWN ARROWS** to move the camera above or below the selected point
    - Hold **CTRL**, then press and hold **THE UP OR DOWN ARROWS** to zoom to toward or away
    - Hold **SPACE**, then use the **ARROW KEYS** to pan the center point
    - Hold **SHIFT** and tap **SPACE** to center the camera on the selected object
    - Keyboard input must be focused on the 3D window. Look for a blue border around the 3D view to indicate that input is routed into this window
- If you have a trackpad or a one button mouse
    - Hold **SHIFT** and move the mouse without pressing the button to orbit the camera
    - Hold **SHIFT AND CTRL** and move the mouse to pan. Do not press the mouse button.
    - Use the keyboard controls to zoom
- The Sandbox software has many buttons, sliders, and various other GUI elements. In this guide, we'll use terminology like **TAB**, **EDITOR**, **PANEL** and **ICON** to refer to GUI elements as shown below.
- 

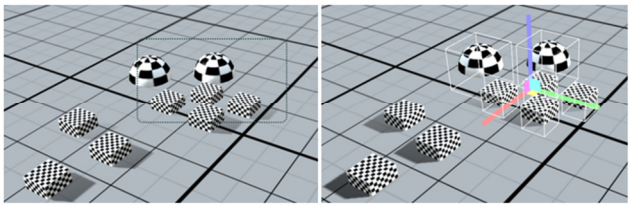
    - A, B – These are **EDITORS**. Multiple **EDITORS** can be open at once
    - C – This is the **CONTENT** Library
    - D – This is the selected **ENTITY.**
    - E – This is a **PANEL**. Each **EDITOR** and the **CONTENT LIBRARY** can have one or more **PANELS**. **PANELS** can be **OPEN** or **CLOSED.**
    - F – These are **TABS**. They open or close **EDITORS**.
    - G – They are the **PLAY**, **PAUSE**, and **STOP** buttons.
- The 3D view of the environment will be referred to as the **STAGE**.
- Every 3D object in the environment is an **ENTITY.**

SELECTING OBJECTS



In order to manipulate objects in the Sandbox, they must be selected. There are many ways to select objects, but only one that we will use today. You must have at least a one button mouse or trackpad for today's exercise. Use your mouse or trackpad to move the mouse cursor over the object you wish to select. Click once with the left mouse button. This will select the object under the mouse. You'll know what object is selected because a bounding box and axis will be drawn on top of the selected object. Many other parts of the GUI will display the selection name as well. This sphere is selected.

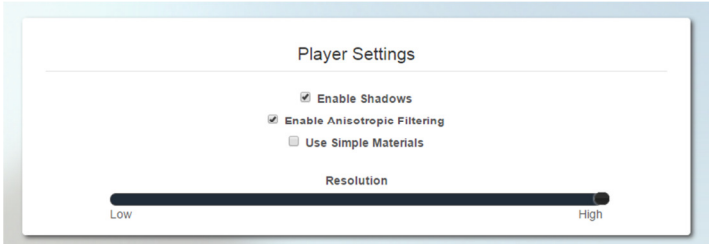
You can select multiple objects at once, though **FOR THIS TUTORIAL YOU SHOULD NOT**. You can select multiple objects by clicking and dragging in the window with the left mouse button. You will see a rectangle drawn on the screen.



It is also possible to select no objects. Simply click once on the ground plane or the sky to select nothing.

PERFORMANCE TROUBLE

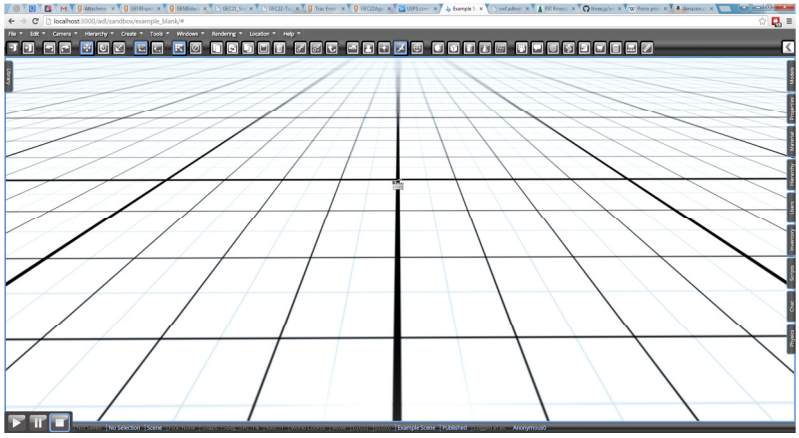
Sandbox is a cutting edge HTML5 application, and can require significant local resources. If you find the software too unresponsive to be usable, please ask for assistance. There are several methods that can be used to reduce the burden on the local computer. If you have trouble running the software, try this:

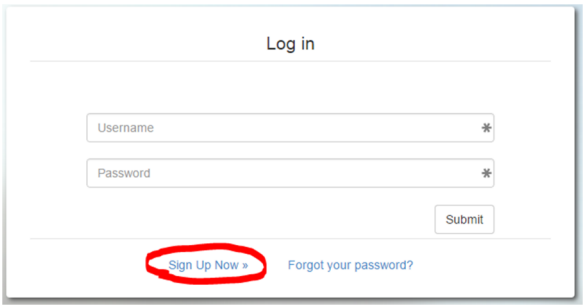
1. Navigate to the homepage of the server you're using.
2. Click the **DEMOS** button.
3. Click the **TOOLS** link on the top right
4. Click the **PLAYER SETTINGS** link
5. Uncheck **ENABLE SHADOWS**
6. Uncheck **ENABLE ANISOTROPIC FILTERING**
7. Check **USE SIMPLE MATERIALS**
8. Drag the **RESOLUTION** slider down toward the middle
9. Try to load the world again. IF you still have performance trouble, try moving the **RESOLUTION** slider all the way to the bottom
10. 

WORKING IN PAIRS OR GROUPS

SETUP AND DESIGN

Before the Sandbox software will allow you to author an environment, you must create an account on the server. For today's exercise, we have allocated a new server with a blank database – this database will be cleared at some point in the future. It is not necessary that your account on this server be accurate. You may choose to list any name or email address you like. Please remember your password – we won't have time during the session to reset accounts.

1. First, let's verify that you can connect to the servers
    a. Navigate your browser to one of the MarsGame servers. The addresses of the 3 servers we have set up for this session are
        i. marsgame1.adltruegame.ch-geni-net.instageni.maxgigapop.net:3000
        ii. marsgame2.adltruegame.ch-geni-net.instageni.maxgigapop.net:3000
        iii. marsgame3.adltruegame.ch-geni-net.instageni.maxgigapop.net:3000
    b. **DO NOT** log in nor create an account.
    c. Click "**CREATE**" on the main page. When prompted choose "**NOPE, I JUST WANT TO TRY THIS**"
    d. You will be assigned to a new blank **TEMPORARY** world.
    e. 
    f. Verify that you're webpage looks something like the image above. Use the information in the reference section to move the camera. If you see the grid like above, and movement looks fluid, congratulations! You meet he hardware requirements.
2. Create a user account
    a. Return to the home page
    b. Click "**SIGN IN**"
    c. If you previously created a user name and password, you may log in now
    d. Otherwise, click "**SIGN UP NOW >>**" at the bottom left

e. 

f. Fill out the required information. You may supply a fake email address if you prefer.
g. Passwords must be at least 8 characters, and contain a capital letter, number, and lowercase letter.
h. Click "**CREATE ACCOUNT**"

3. Create a new world
   a. From the home page, click the large blue and white **CREATE** button
   b. Choose a memorable title, and click **CREATE.** On the next window choose **LAUNCH**.
   c. You should now enter an environment similar to that in step 1E. This environment however is owned by you and will persist on the server until you explicitly delete it.

EXECUTE:

Now that we have an account and a space to work, we can get down to designing our game environment. I've prepared the server with a content library that includes all the artwork from the Mars Game. The next steps will guide you through building a simple game based on the available assets. The goal of the game will be to navigate a rover through terrain without hitting any obstacles while picking up objects.

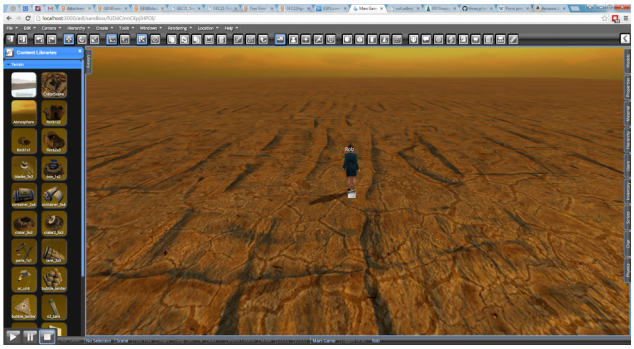
Please ask if you wish to learn how to import additional objects, object libraries, or how to create new objects and libraries. The Sandbox toolkit has many features and functions that we will not cover today – please wait until you are finished with this exercise to explore these – it may be possible to accidently break your environment if you're not careful with some of these other features. While I can reset your environment from a database backup, I may not have time to assist you during the session.

1. Setup the environment
   a. Our environment is pretty bleak – let's pull out some background graphics from the library. To access the library, look for a **TAB** labeled **LIBRARY** on the left of the screen. It will look something like this:
   b. 

   c. Click this **TAB**, and the **CONTENT LIBRARY** will slide open. The **CONTENT LIBRARY** is preloaded with a set of content organized into groups. Each group has a **PANEL** that can be clicked to expand. The **TERRAIN** tab is already open.
   d. Find the **ATMOSPHERE** icon. Drag this icon onto the **STAGE** and drop it.

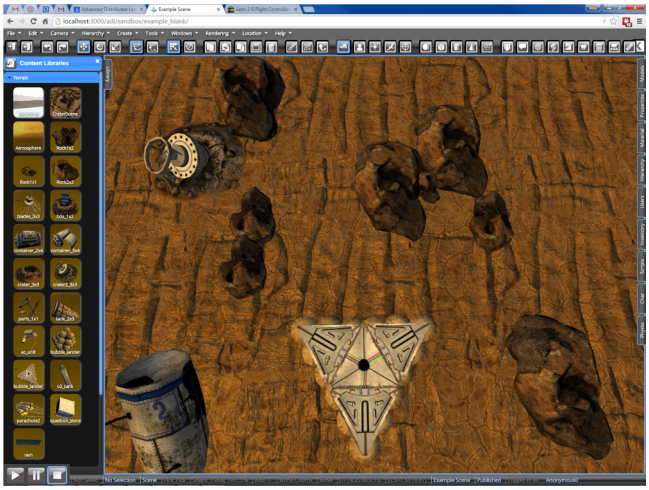
e. 

f. You'll notice a big change right away. Your white grid and sky are replaced by a broken rock texture and a red cloud background

g. 

h. Now, let's create a bit more background. While still in the **TERRAIN** panel of the **CONTENT LIBRARY**, find the **BACKDROP** icon and drop it into the **STAGE** in a similar way. You'll see mountains appear in the distance.

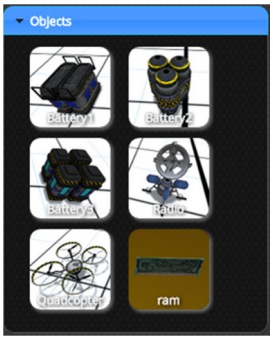
i. There are many other icons in the **CONTENT LIBRARY** that you can add to the **STAGE**. Take some time to build a simple environment for your rover to navigate. I recommend at least creating a few rocks and the "bubble_lander", but feel free to be creative. Here's my scene for your reference:

j. 

k. Be sure to create **ONE ENTITY** from the **PANEL** labeled **ROVERS.** Place the **ENTITY** from the **PANEL** labeled **ROVERS** in a good location for the game to start

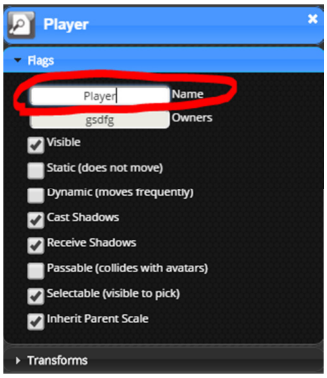
l. 

m. Create at least **ONE ENTITY** from the **PANEL** labeled **OBJECTS**.

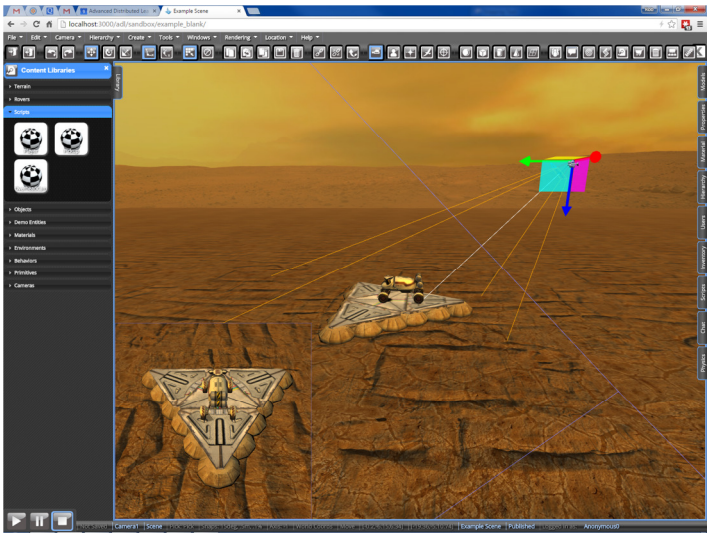
n. 

o. From now on, we will refer to the **ENTITY** that you created from the **PANEL** labeled **ROVERS** as the **ROVER**

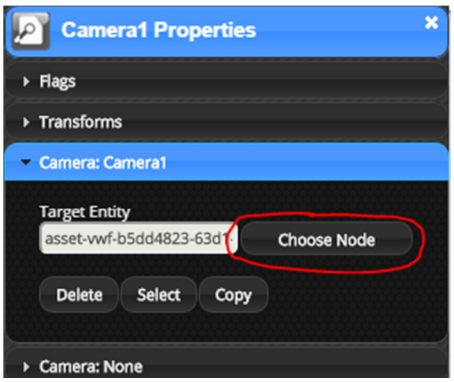
2. Add the scripts
   a. Now that the environment is set up, we can start adding the scripts that will create interactivity. I've created most of the code ahead of time. You'll find each behavior you need under the **PANEL** labeled **SCRIPTS** in the **CONTENT LIBRARY**
   b. Drag and drop the **PLAYER** icon from the **PANEL** labeled **SCRIPTS** to your **ROVER**. You can check that you created it properly by selecting the **ROVER**, and opening the **HIERARCHY EDITOR**. If you see **BEHAVIOR1** under **VWF OBJECT CHILDREN** then you've added it successfully.
   c. At this point, the game will do something when executed! Hit the **PLAY BUTTON** in the bottom left corner, then click in the **STAGE** and use the **WASD** keys to drive the **ROVER**. The script is built to move the rover on the +Y direction in the world with the W key, so lay out your level appropriately.
   d. 
   e. Click the **STOP BUTTON**. Now that the **ROVER** has the script attached to it, let's set a name for the **ROVER**. This name will help the other scripts in the game understand which **ENTITY** the user is controlling. Select the **ROVER**. Open the **PROPERTIES EDITOR**, then find the **NAME** field and replace the existing text with "Player".

f. 

g. From the **CONTENT LIBRARY** select the **PANEL** labeled **SCRIPTS** and drag and drop the **OVERHEADCAM** script into the scene. This script will create a new , invisible **ENTITY** that tracks the **ROVER.** When this **ENTITY** is selected, you will see a preview in the bottom left of the **STAGE.**

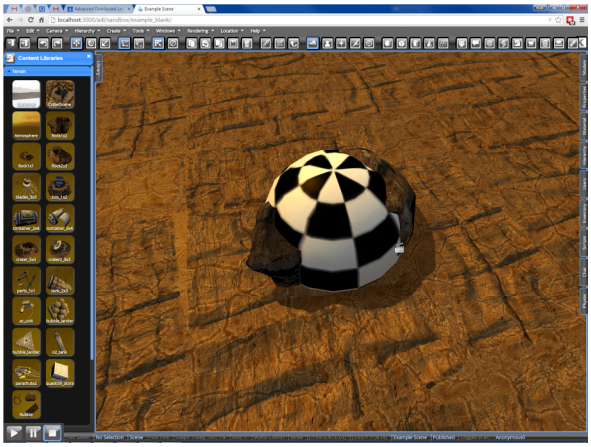
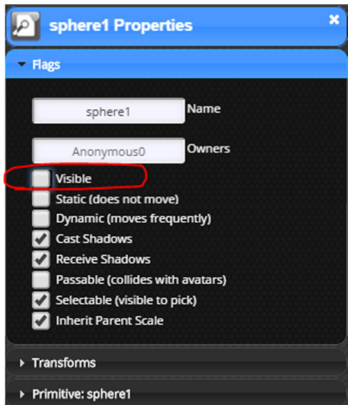
h. 

i. You can position this **ENTITY** as you like, but it won't matter – the scripts will move the camera to follow the **ROVER** automatically. Open the **PROPERTIES EDITOR** (while the new camera **ENTITY** is selected), then find the **TARGET ENTITY** field. Click **CHOOSE NODE** and then click on the **ROVER**. The overhead camera **ENTITY** is now set to track the **ROVER**. You can see this if you click the **PLAY BUTTON** again.

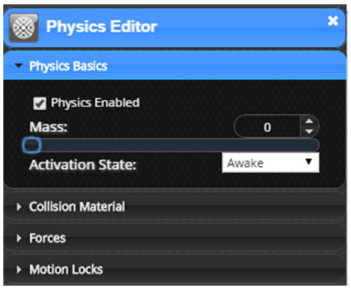
j. 

k. You can activate this camera as the main camera by selecting the **CAMERA->ACTIVATE CAMERA** pull-down menu option and choosing the **CAMERA1** option. You can switch back by choosing **EDITOR CAMERA** from this menu.

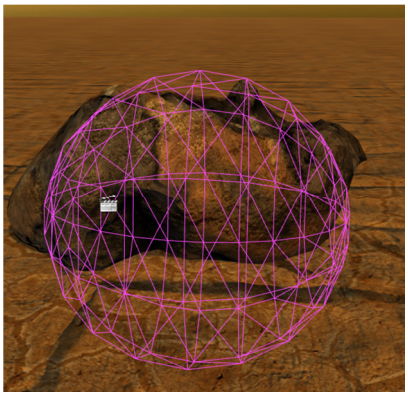
l. Finally, from the **CONTENT LIBRARY** select the **PANEL** labeled **SCRIPTS** and drag and drop the **PICKUP** icon onto the **ENTITIES** you created in step 1M.
m. These **ENTITIES** will from now on be referred to as **PICKUPS**
n. Test your game by clicking the **PLAY BUTTON** – if everything is working, you can drive over these **PICKUPS** and they will disappear.

3. At this stage, we have a basically working game, but there are no failure conditions and no real success either. If you're a bit behind, you can skip the rest of the **EXECUTE** section. The rest of the tutorial will still work. Now, we're going to get fancy. We are going to setup a physics simulation that will inform us when the **ROVER** intersects a rock or obstacle. This information will come in the form of a callback that we will handle in code.
    a. We'll create a new **ENTITY** to collide with the **ROVER**. Open in the **PANEL** in the **CONTENT LIBRARY** called **PRIMITIVES**. Drag and drop an **ENTITY** from this **PANEL** into the **STAGE**. Place it over an **ENTITY** that the **ROVER** should not collide with.
    b. You can change the length, width, or radius of this new **ENTITY** by opening the **PROPERTIES EDITOR** and manipulating the **LENGTH**, **WIDTH** or **RADIUS** sliders. Each type of **ENTITY** has different sliders. **DO NOT USE THE RESIZE TOOL**. Notice how I place this sphere over the rock?
    c. This sphere will be invisible, but cause a collision, so we call it a **PROXY** for the rock.
    d. 
    e. While the **PROXY** is selected, open the **PROPERTIES EDITOR** and under the **FLAGS PANEL**, uncheck the **VISIBLE** checkbox. The **ENTITY** will disappear, but remain selected.
    f. 

    g. Open the **PHYSICS EDITOR**, and check the **PHYSICS ENABLED** checkbox. Make sure to set the **MASS TO 0**. This ensures that the collision shape cannot move when the **ROVER** impacts it.

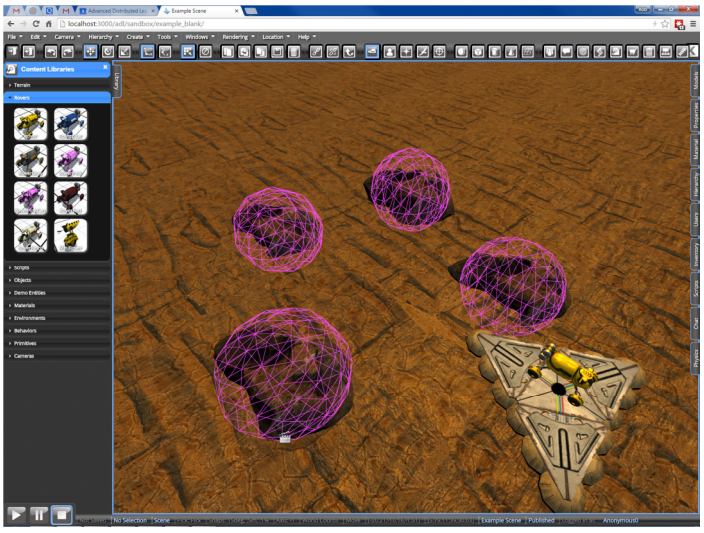
h. 

i. You will see a pink outline of the collision shape, even though the **ENTITY** is invisible. This may render behind or in front of the object – it does not matter which.

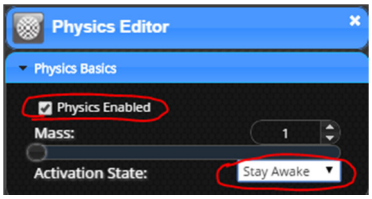
j. 

k. This pink preview is visible only when the **ENTITY** is selected – you can activate a persistent physics view for all collision **PROXIES** by selecting **RENDERING->TOGGLE PHYSICS DISPLAY** from the top menu bar.

l. Set up collision **PROXIES** for each object you wish the **ROVER** to collide with. You can duplicate your existing **PROXY** with copy/paste, or the **DUPLICATE** command from the top **EDIT MENU** . This will make the process significantly faster. To save time, you might only setup one **PROXY** for demonstration's sake.

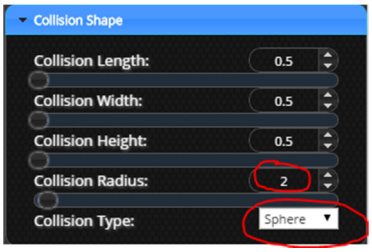
m. 

n. Now, we must make the **ROVER** part of the physics simulation. Select the **ROVER** an open the **PHYSICS EDITOR**

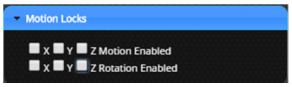
o. 

p. Check **PHYSICS ENABLED**, and **ACTIVATION STATE** to **STAY AWAKE**.

q. Under the **COLLISION SHAPE PANEL**, set **COLLISION TYPE** to **SPHERE** and **COLLISION RADIUS** to **2**

r. 

s. Finally, under the **MOTION LOCKS PANEL**, uncheck all the checkboxes

t. 

u. Now, we've setup the physics simulation to cause a physics **PROXY** to follow the **ROVER ENTITY**, but to not allow the simulation to move the **ROVER**. Instead, we will just use the collision shape to test for intersections.

4. Next, we create a screen that will appear when the user collides with a collision **PROXY**. This screen will be a 2D image the pops up over the screen.

a. First, **MAKE SURE NOTHING IS SELECTED**, then from the **TOP MENU**, choose **CREATE->GUI ELEMENTS->IMAGE**. You will see an icon appear in the center of the screen. This icon may be blank, or display a "missing texture" image. Select it by clicking on it.

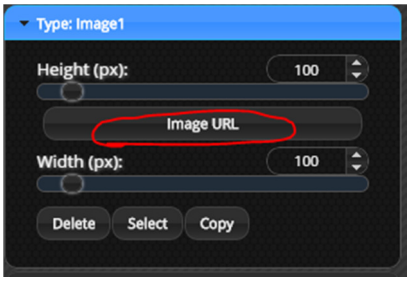
b. This is the blank **IMAGE ENTITY**.

c. 

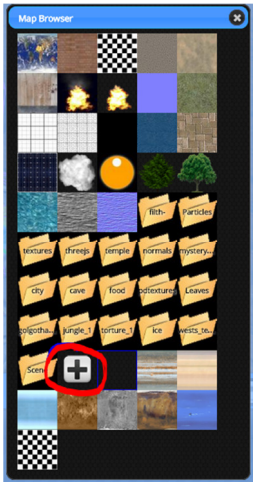
d. When it is selected, it has a blue border

e. 

f. In the **PROPERTIES EDITOR**, find the **IMAGE URL BUTTON** on the **PANEL** labeled **TYPE:IMAGE1**. This will let us pick the image that is displayed.

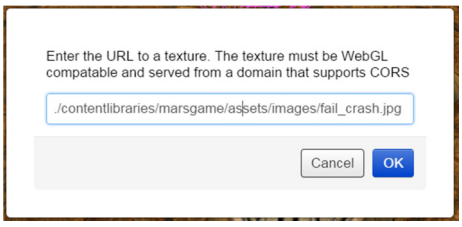
g. 

h. When you click this button, the **MAP BROWSER** will appear. Find and click the **PLUS ICON** in this list of images.

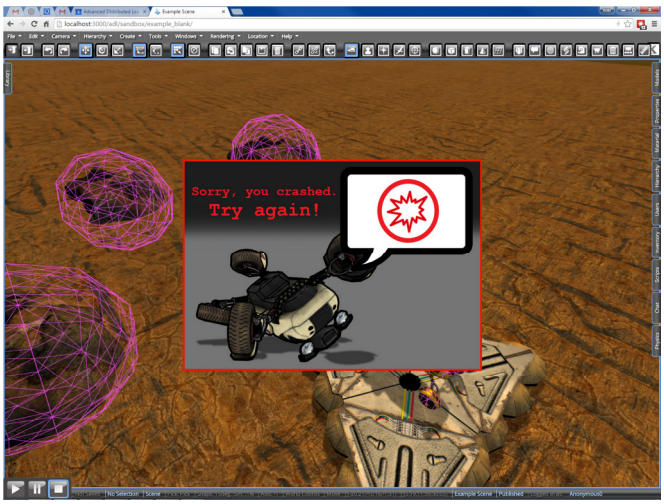
i. 

j. This will pop up a prompt for you to enter the URL to an image. Enter
./contentlibraries/marsgame/assets/images/fail_crash.jpg

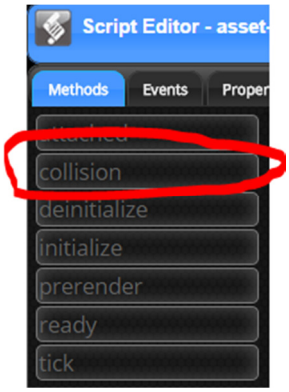
k. Click **OK**, and close the **MAP BROWSER** by clicking the **X** in the top right corner.

l. 

m. You can use the **WIDTH** and **HEIGHT** sliders to change the size of the **IMAGE ENTITY**, and place it on the screen with the first 2 **TRANSLATION** boxes in the **TRANSFORM PANEL**. Place it center **STAGE**.

n. 

o. While the **IMAGE ENTITY** is still selected, find the **VISIBLE** checkbox in the **FLAGS PANEL** of the **PROPERTY EDITOR**, and toggle it until the picture disappears.

5. Finally, we're going to write a bit of code to make this box appear when the **ROVER** intersects a collision **PROXY ENTITY**.
   a. Select the **ROVER**, then open the **SCRIPTS EDITOR**. You'll find this in a **TAB** on the right.
   b. Click the **COLLISION** button on the **METHODS TAB**. This is a suggestion to you – the engine knows this **ENTITY** might get collision notifications, and is prompting you to handle them with code on this method. When prompted, click **OK**.
   c. 

   d. In the body of the function, you'll see a place you can type. Enter this code exactly as shown here. The red border indicates that the code has not been saved.

```
function collision ()
{
        if (arguments[0]) this.Scene.findNode('Image1').visible = true;
}
```

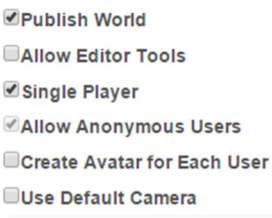
   e. Click the **SAVE METHOD** button. You can now deselect the object and test the game.
   f. If you've still got time in the session, congrats! You're doing great. I'll leave generating a success screen and the success criteria as an exercise for the reader. Here's the basic outline:
      i. If there is a **METHOD** on the **ROVER** called **PICKEDUPENTITY**, it will be triggered when the **ROVER** picks up one of the **PICKUP ENTITIES**.
      ii. You can create another **IMAGE ENTITY**, and display it when an **PICKEDUPENTITY** is triggered

iii. If you're feeling very adventurous, you can keep a count of the number of times **PICKEDUPENTITY** has been triggered, and only show this screen when all the **ENTITIES** are picked up.

FINISH:

Now that you have a working game with a goal (picking up the objects) and a challenge (avoiding the rocks), you can play it, and publish it out for others to use. You've probably been testing all along, but test one more time in the editor. When you're satisfied that everything is working, you can just leave the webpage, and navigate to the Sandbox homepage. From here, we can publish the world and share it for others to play. The world will be saved automatically.

1. Navigate back to the home page.(**GENI SERVER URL**)
2. Find your world. You can click **WORLDS** then **MY WORLDS**. Locate your world and click it so you see the landing page as in step **SETUP:3B**
3. Click the **SETTINGS** link.
4. Change the settings like this – we want a published world that is **SINGLE PLAYER**, does **NOT INCLUDE THE EDITOR INTERFACE AND DOES NOT CREATE AN AVATAR**



5.
6. When you uncheck **USE DEFAULT CAMERA,** the **CAMERA** menu will appear. From the **CAMERA** dropdown, choose **CAMERA1**. It should be the only entry in the list. This ensures that the players will only be able to see the world from the view of the camera object we created, instead of the editor camera you've been using.
7. Click **SAVE**.
8. You can now launch the game by clicking the large icon, or the **LAUNCH** button. The game will run in the browser without any tools, and from the camera you have specified. You can share the link with your friends from the **EMBED** link, of just send them the URL of this page.
9. When you're done, you can delete your game from the **DELETE** link. We'll leave them on this server for a while, but eventually we'll tear down the platform. Please email me if you would like me to export your work for you to keep.