Simple Collision Detection

By David Yan

Under the direction of Professor Susan Rodger and Chari Distler

Duke University, June 2015
Part 1: Setting up the scene

- Your first step will be to choose an environment for your world.
- We will be working with a sand world in this tutorial.
Part 1: Setting up your scene

- For this world, we want to first add a **Humvee** object. To get the **Humvee**, go to **Add Objects>Vehicles** and then select the **Humvee** from the Vehicles folder.
- Position it in your world like so →
Part 1: Setting up your scene

• Now go to the **Buildings** folder and add a **Windmill**, **Factory**, and **Townhouse1** to your world.

• Position the objects so that your world looks like this →

• **It is very important that your Humvee be facing in the direction of the buildings**
Part 2: Adding an event

• Now we want to be able to move our Humvee around the world using the arrow keys. To do this, we need to create a new event in our world.

• In your event space, select **Create new event > Let the arrow keys move subject**
Part 2: Adding an event

• Now that you have that event selected, we want to move our Humvee with the arrow keys, so select that option in your Let arrow keys move subject event.
Part 2: Adding an event

• If you run your world, you will notice that while your Humvee moves with the arrow keys, the camera view does not change even when your **Humvee** moves off screen.

• To have the camera follow the Humvee as it moves around the world, we want to change the vehicle setting of the camera itself.

• Change the **camera’s vehicle** from World to **the entire humvee**
Creating a List

• Now we are going to create a list of all the building objects. Click on World in the object tree and under the properties tab of the world, select create a new variable. We want to create a list of objects. Once this option is selected, simply add the objects to the list like so
Part 3: Creating the Method

• We want to now write a method so that when the Humvee runs into a building, or collides with a building, the Humvee will stop and move back upon collision.
• To begin, select the World in your object tree and then select the methods tab.
• Now create new method in the World’s method tab and call it “collision.”
Part 3: Creating the method

- With your `world.collision` method tab open (it should be empty at this point since we have not added any code), drag in an **If/Else** statement.
- Your method should look like this:
Part 3: Creating the method

• To test whether a collision has occurred, we want to test whether the Humvee is within a certain distance of a building. For this, we will use a Humvee function.

• Click the function tab under Humvee in the object tree, select **Humvee is within threshold of object**, and drag it over the true block in the If/Else statement.

• Finally, select **15 meters** for the distance, and select **factory** for the object.
Part 3: Creating the method

- Your world.collision method should look like this so far:
Part 3: Creating the method

• Now, if the Humvee is within this distance of the factory, we want the Humvee to bounce, or move, backward one meter very quickly.

• In the If/Else block, where the first “do nothing” is, drag up a Humvee move block and select backward and then 1 meter. Since we want to do this quickly, we want to set the duration to 0.01 seconds.
Part 3: Creating the method

• Now 15 meters is an arbitrary number and we want our threshold calculation to be as realistic as possible, so we are going to change “15 meters” to half of the Humvee’s width.

• To access the Humvee’s width, select Humvee in the object tree and go to the properties tab.
Part 3: Creating the method

• To complete our If/Else block, we will now **add half the width of the factory** to **half the width of the Humvee** using the math options found by clicking the gray arrows.
Part 3: Creating the method

• Your World.collision method should look like this so far:
Completing your world

• Instead of being restricted to only checking for collisions with the factory, we can use a For Loop to check for collisions with every object in our list.

• To complete your world.collision method, drag in a For all together block. Since we want the loop to go through our list of buildings, be sure to select your list of buildings.

• Your method should look like this so far:
Completing your world

- Finally, drag in your If/Else block into the For loop and replace all (2) instances of **factory** with **item_from_building**. This way, we are able to check every single building in the list for a collision instead of just the factory.
Adding a final event

• Finally, we want to create an event such that while our world is running and we are moving our Humvee, we want the method to check for collisions.
Final Step

• Now that you have a “While the world is running” event, change the “during” phase of that event to world.collision.

• Your event space should look like this: