Minetest Mods

Skills/Concepts

This module guides students through creating a mod for a Minetest tool. It will cover some Lua scripting using the Minetest Application Programming Interface (API).

The following concepts will be touched upon in this module:

- console use
- basic usage of Minetest
- modding

Console Use

This module relies heavily on the command-line. You will be using a terminal emulator to execute the following commands.

Command	Description
cd	Change directory
ls	List directory contents (whats inside)
subl	Sublime text editor

Overview

Modules for Minetest require a specific directory hierarchy. The files consist of an entry-point script for initialization and supporting files. Generally the supporting files consist of additional scripts, PNG files, documentation, etc. Although Minetest is written in C++, the scripting language uses Lua. Minetest provides hooks for sophisticated eventing and other application programmer interfaces (APIs).

Modules for Minetest require a specific directory hierarchy. The files consist of an entry-point script for initialization and supporting files. Generally the supporting files consist of additional scripts, PNG files, documentation, etc. Although Minetest is written in C++, the scripting language uses Lua. Minetest provides hooks for sophisticated eventing and other application programmer interfaces (APIs).

We will be adding chat command to Minecraft which will trigger a change in gravity in the game. We will allow for the player to specify how much gravity they want with the command.

Process

Create the Base Module

The commands in this section should be executed from a terminal in the `minetest/mods` directory.

0. Create a directory to hold the module.

   mkdir lesson_2

1. Create the entry-point script, `lesson_2/init.lua`, and put the following text in it:

```
minetest.register_chatcommand("lesson_2:antigravity",
    {
        params = "",
        description = "Lesson 2: Antigravity enabler",
        func = function(name, param)
                local player = minetest.get_player_by_name(name)
                if not player then
                  return false, "Player not found"
                end
                player:set_physics_override(
                {
                  gravity = 0.2 -- set gravity to 20% of its original value
                                -- (0.2 * 9.81)
                })
        end
    })
```

2. Run the game, create a new Local Game.
3. Configure the new local game to enable the custom module.
4. Enter the game (in creative mode).
5. Type `/antigravity` and press Enter.
6. Jump.

Make it Configurable

Alright, we can lower gravity and jump like Superman... but what if we get bored of 20% gravity? We would have to quite the game, modify gravity in the 'init.lua' script to a new value. Lets modify the script so we can change the value of gravity in-game!

0. Edit the `lesson_2/init.lua` file and put the following in it:

```
minetest.register_chatcommand("lesson_2:antigravity",
    {
        params = "<percent>",
        description = "Lesson 2: Antigravity enabler",
        func = function(name, param)
                local player = minetest.get_player_by_name(name)
                if not player then
                  return false, "Player not found"
                end
                player:set_physics_override(
                {
```

```
            gravity = param -- set gravity to 'param'% of its original value
                            -- ('param' * 9.81)
          })
    end
})
```

1. Run the game, create a new Local Game.
2. Configure the new local game to enable the custom module.
3. Enter the game (in creative mode).
4. Type `/antigravity 20` and press Enter.
5. Jump.
6. Uh oh...

What Went Wrong?!?!

We tried to jump but barely left the ground, what happened??? Take a few minutes and see if you can figure it out before we fix the mistake.

The problem is with our math. We want the player to what percent of gravity they would like to experience. However, the number the player provides is not a percent but the multipler of gravity. By typing in '/antigravity 20' we actually increased gravity by 20x.

To make this into a percent, we need to divide the number the player provides by '100'. Make the following modification to the script to fix the problem:

0. Edit the `lesson_2/init.lua` file and put the following in it:

```
minetest.register_chatcommand("lesson_2:antigravity",
    {
        params = "<percent>",
        description = "Lesson 2: Antigravity enabler",
        func = function(name, param)
                local player = minetest.get_player_by_name(name)
                if not player then
                  return false, "Player not found"
                end
                player:set_physics_override(
                {
                  gravity = param / 100 -- set gravity to 'param'% of its original value
                                        -- ('param' * 9.81)
                })
        end
    })
```

1. Run the game, create a new Local Game.
2. Configure the new local game to enable the custom module.
3. Enter the game (in creative mode).
4. Type `/antigravity 20` and press Enter.
5. Jump.

Conclusion

First, we created very simple player modification to change the gravity. We accomplished this by using the `minetest.register_chatcommand()` API function to create a new chat command and define what happens when it is envoked. The first iteration of our script had a static modifier to the gravity that could not be changed in game.

The second iteration of our script allowed the player to specify the desired amount of gravity in game by providing a number after the command (`/antigravity 20`). While this modified the gravity, it did not decrease it like we expected.

After reviewing the code, it became apparent we were multipying gravity by the number provided by the user instead of using the number as a percent. We were able to fix the code to convert the user provided number into a percent by dividing it by `100`.