

DV21566

# The Next Step in Design Visualization

Ben Bisares

Autodesk – Media & Entertainment Technical Specialist

## Description

Learn how to harness the power of Live Design to bring your creations to life. In this class we will show you how to take your Revit content and create real-time interactive presentations using either the Live or 3ds Max and Stingray workflow and see which one works best for you. With Live Design ecosystem we will see how to create content with features like real-time lighting, animation, object interaction, and even virtual-reality headset support. Last, we will learn how to create a stand-alone package compatible with common platforms like Microsoft Windows, Google Android, and Apple iOS. This session features Autodesk Revit, Live Editor, Stingray and 3ds Max. AIA Approved

## Learning Objectives

- Live Design Visualization workflows overview
- Live Design vs. “traditional” workflow
- Preparing assets for Stingray using 3ds Max
- Creating and packaging Stingray projects for Windows/Android/iOS

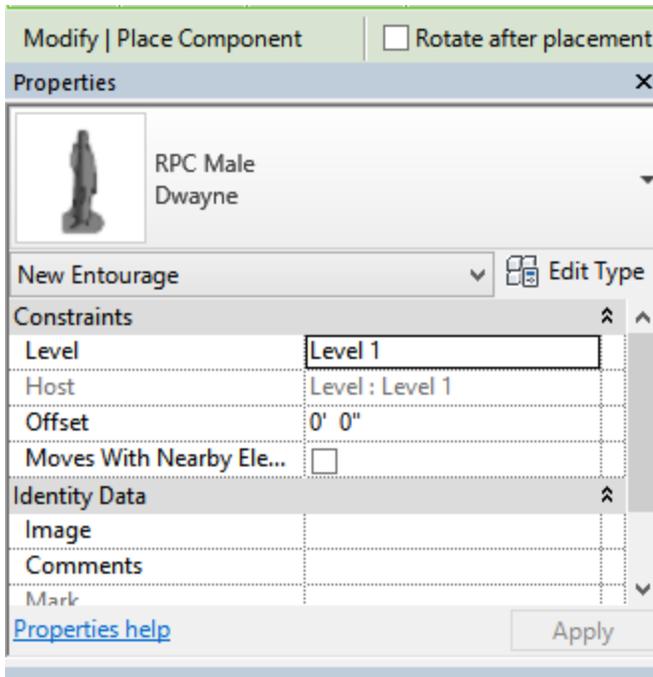
## Your AU Expert

Ben is a Media & Entertainment specialist at Autodesk. With over 10 years of experience in both the Video Game and Architecture industry. Always excited to learn the potential of emerging technologies he has worked on multiple projects using Virtual Reality, 3D printing, and the Arduino platform.

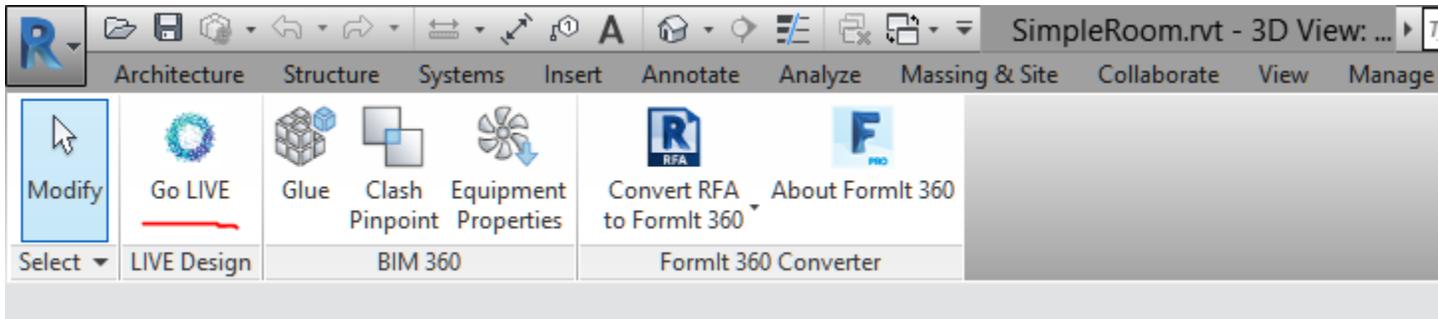
## Where to find the Entourage RPC assets

[https://apps.autodesk.com/RVT/en/Detail/Index?id=4500868201744478807&appLang=en&os=Win32\\_64](https://apps.autodesk.com/RVT/en/Detail/Index?id=4500868201744478807&appLang=en&os=Win32_64)

Although the Entourage pack contains boats, vehicles, furniture, and men & women, the LIVE Editor is currently only compatible people and foliage content



## Creating a LIVE Design project in Revit



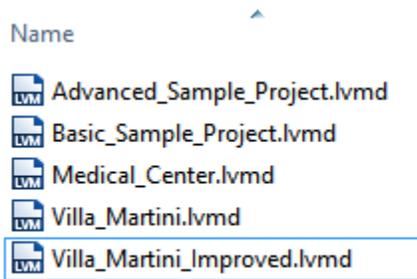
Must be in a 3D view

Detail level must be set to fine



This is a cloud service so you must be connected to the internet

## Bringing LIVE Design models into Stingray

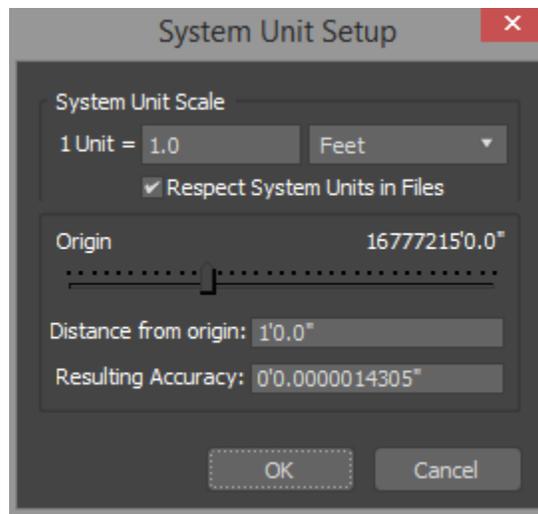


\*.lvmd files are simply an archive of a Stingray project and its associated files. You can make a copy of the \*.lvmd file and rename the extension to .zip to allow you to unpack the files using something like Winzip or 7-Zip

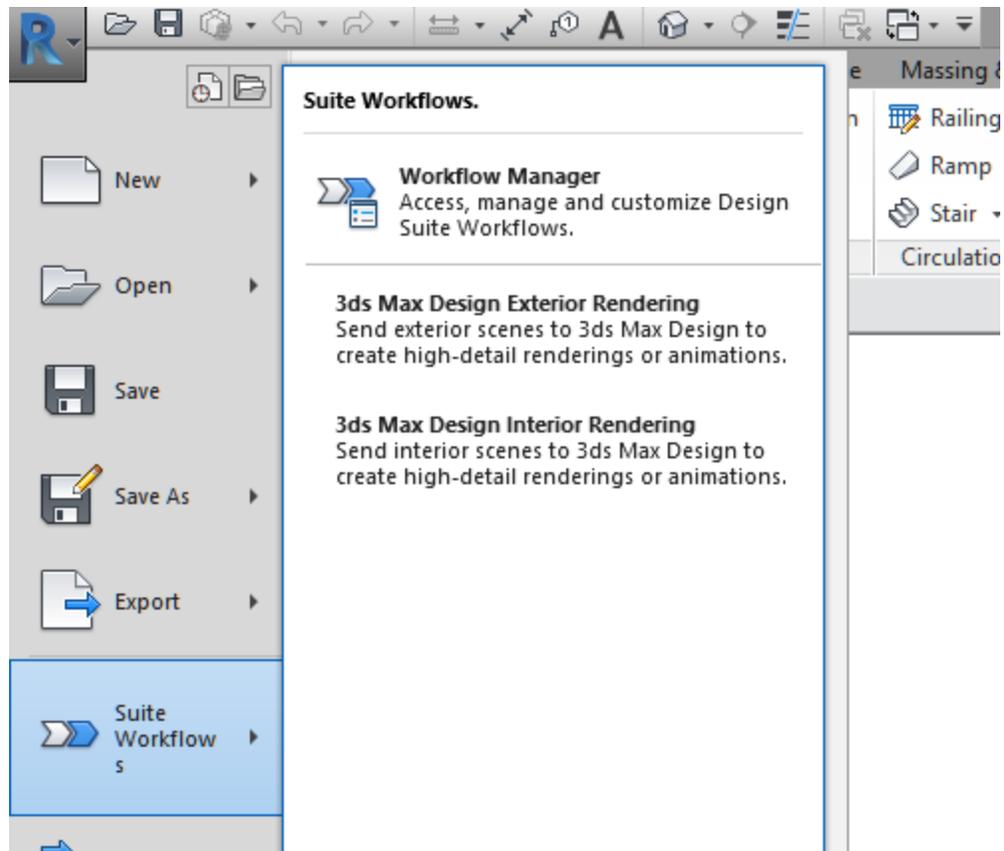
## Revit to 3ds Max

First ensure that system units are set to feet in 3ds Max to ensure proper scaling when importing from Revit

-Customize > Unit Setup > System Unit Setup



In Revit use the "Suite Workflow" tool to send your Revit project to 3ds Max



## Working on Stingray projects in 3ds Max

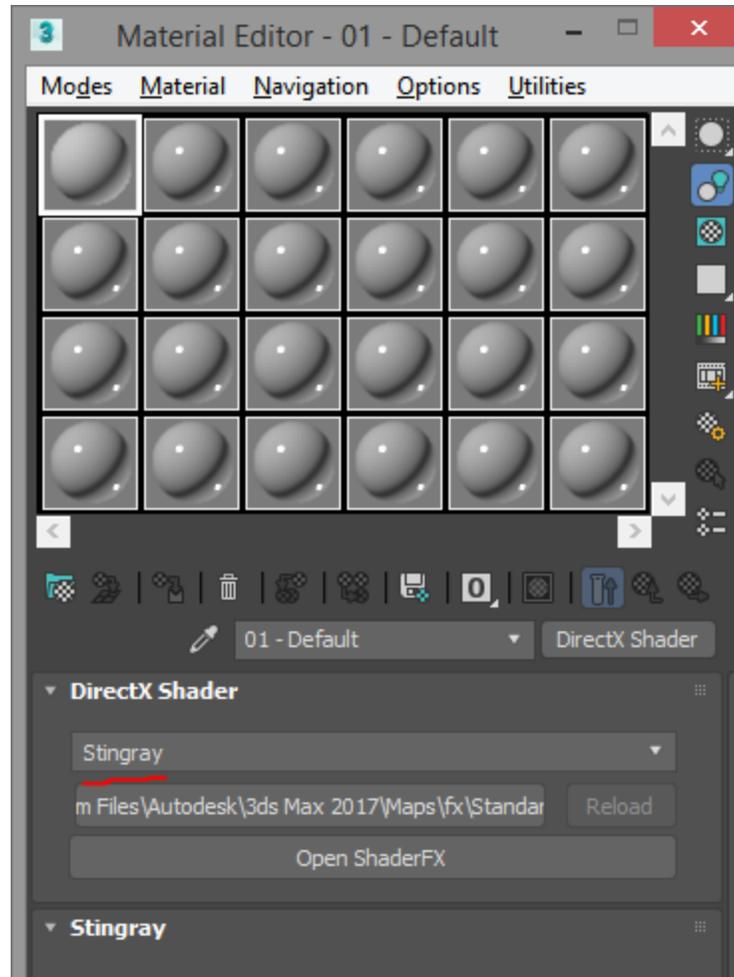
### Where should I work on the materials?

It's best to work upon materials in 3ds Max rather than in Revit as only 3ds Max will give you an accurate representation of the final result

### What material should I use?

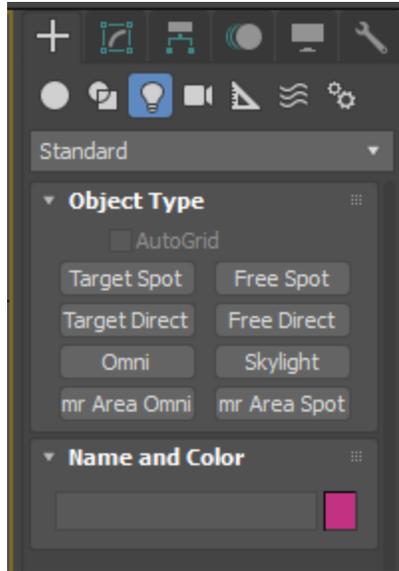
Although Stingray is compatible with certain materials in 3ds Max, it's best to use Stingray DirectX shaders to ensure material consistency between programs as all materials will end up as a Stingray shader eventually. In addition using DirectX shaders in 3ds Max will allow you to see the same result in the viewport as you would see in Stingray

-Material Editor > Material/Map Browser > DirectX Shader > Choose "Stingray" from the drop down



## Lighting

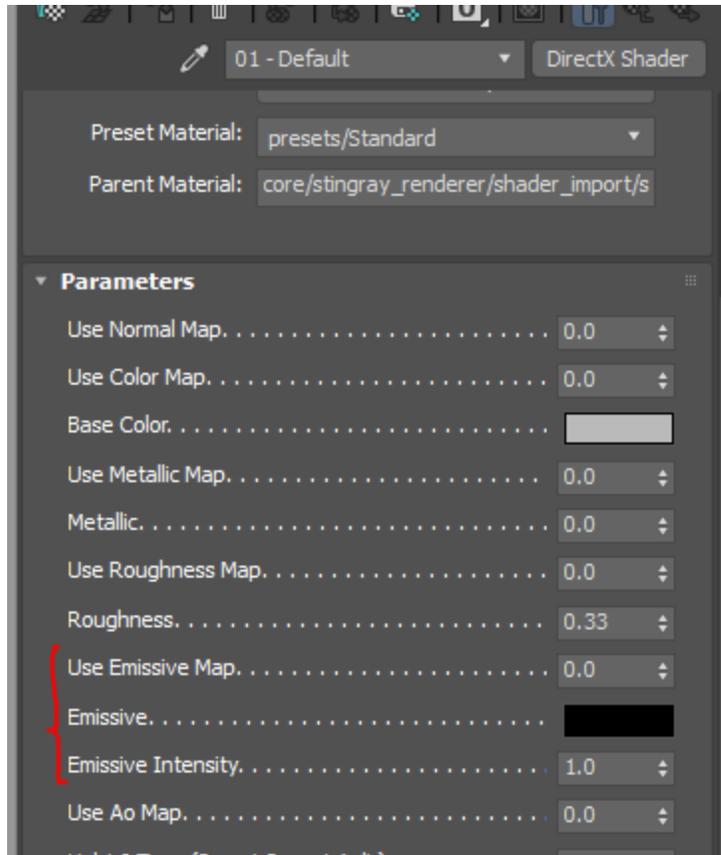
Since Stingray based on game engine technology many physicality based attributes cannot be calculated. Photometric lights should not be used as they cannot be properly handled in game engines. Use standard lights in 3ds Max to ensure that you don't get any surprises later on.



If your scene contains photometric lights that came from Revit it's best to replace them now rather than in Stingray later.

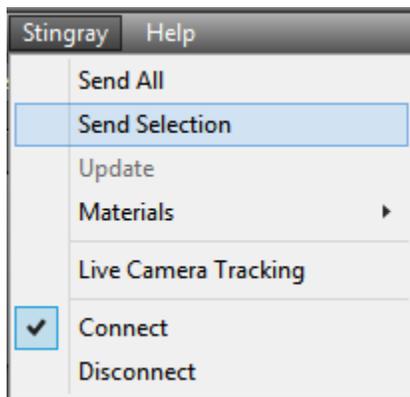
## Emissive shaders

Stingray supports emissive shaders. Emissives allow you to create a glowing like effect on a shader. This also behaves like an area light source and is one way to simulate large (ex: fluorescent panels) or irregularly shaped lights (ex:neon lights)



## Exporting to Stingray

Exporting to Stingray is a simple process of ensuring that Stingray is running with the target project open. In 3ds Max you can then use the Stingray “Send” command to send the assets to Stingray



Separately or together?

You could select multiple objects and send them as one unit to Stingray provided that the following criteria are met:

- All materials/shaders have already been assigned to your satisfaction. It is not feasible to assign a new/different shader to part of a unit once in Stingray
- You do not intend to move/rotate/scale part of a unit later on

Generally this means that structure type objects (floors, walls, ceilings) are okay to export as one unit but including furniture that would be desirable to be moved relative to said structure is not a good idea

## Stingray Templates

For architectural or product design projects you generally want to use one of four default templates in Stingray

-Basic, Empty, VR\_oculus, VR\_Steam

For non-VR projects use basic or empty. The basic template contains a prebuilt menu, player camera, and HUD. The empty project contains the bare minimum and will require you to add elements yourself.

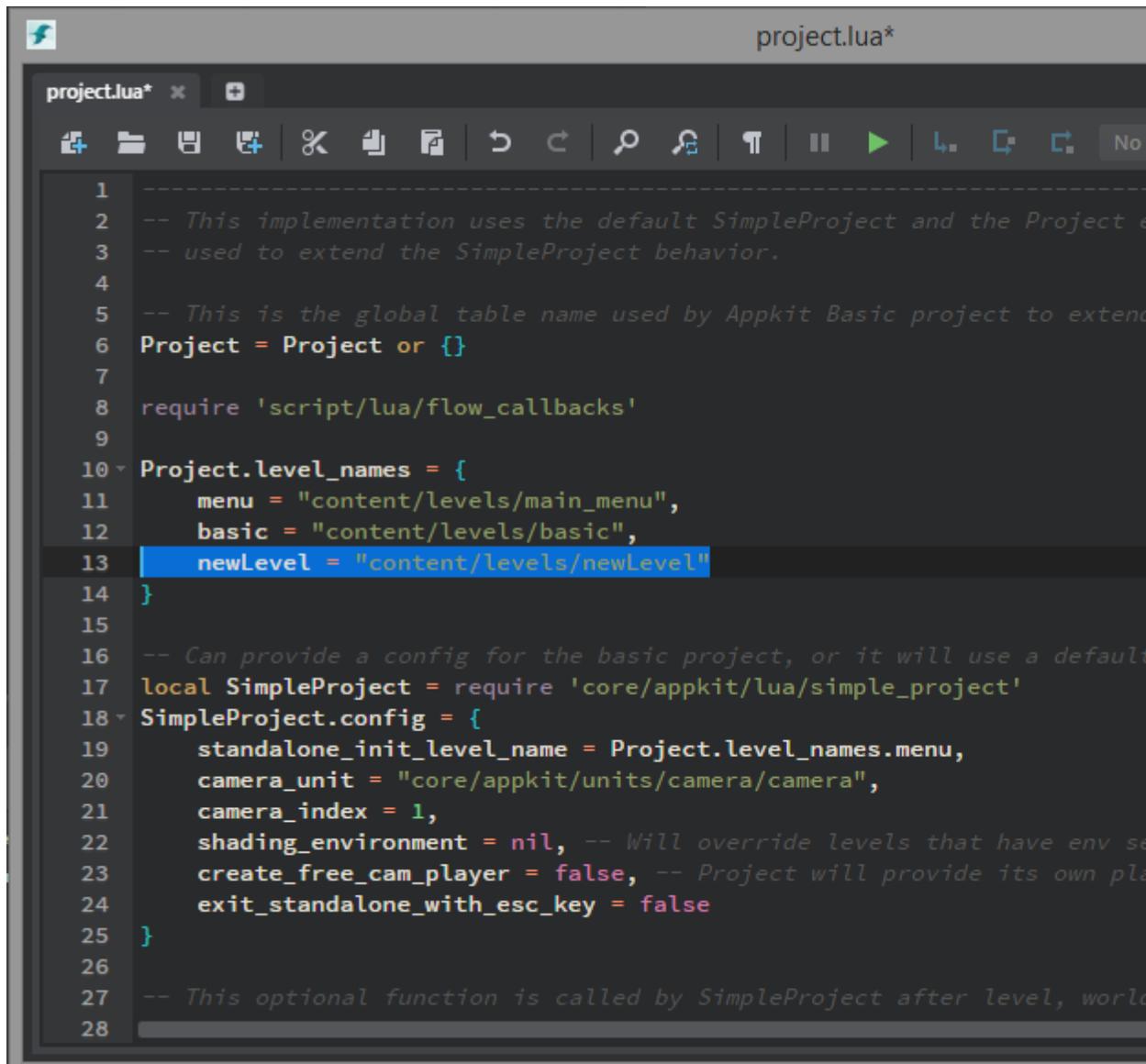
For VR projects use the template for the device that you are targeting (Oculus Rift or HTC Vive). These templates contain prebuilt modules (controller, HMD, UI elements, etc.) for their respective devices. Since these templates already contain some content (exception being the empty template) you want to create a new level or delete the contents of the current level for a fresh start.

Note that if you are creating a new level you must modify the files below with the name of your new level if the project is to work correctly:

### Basic Template

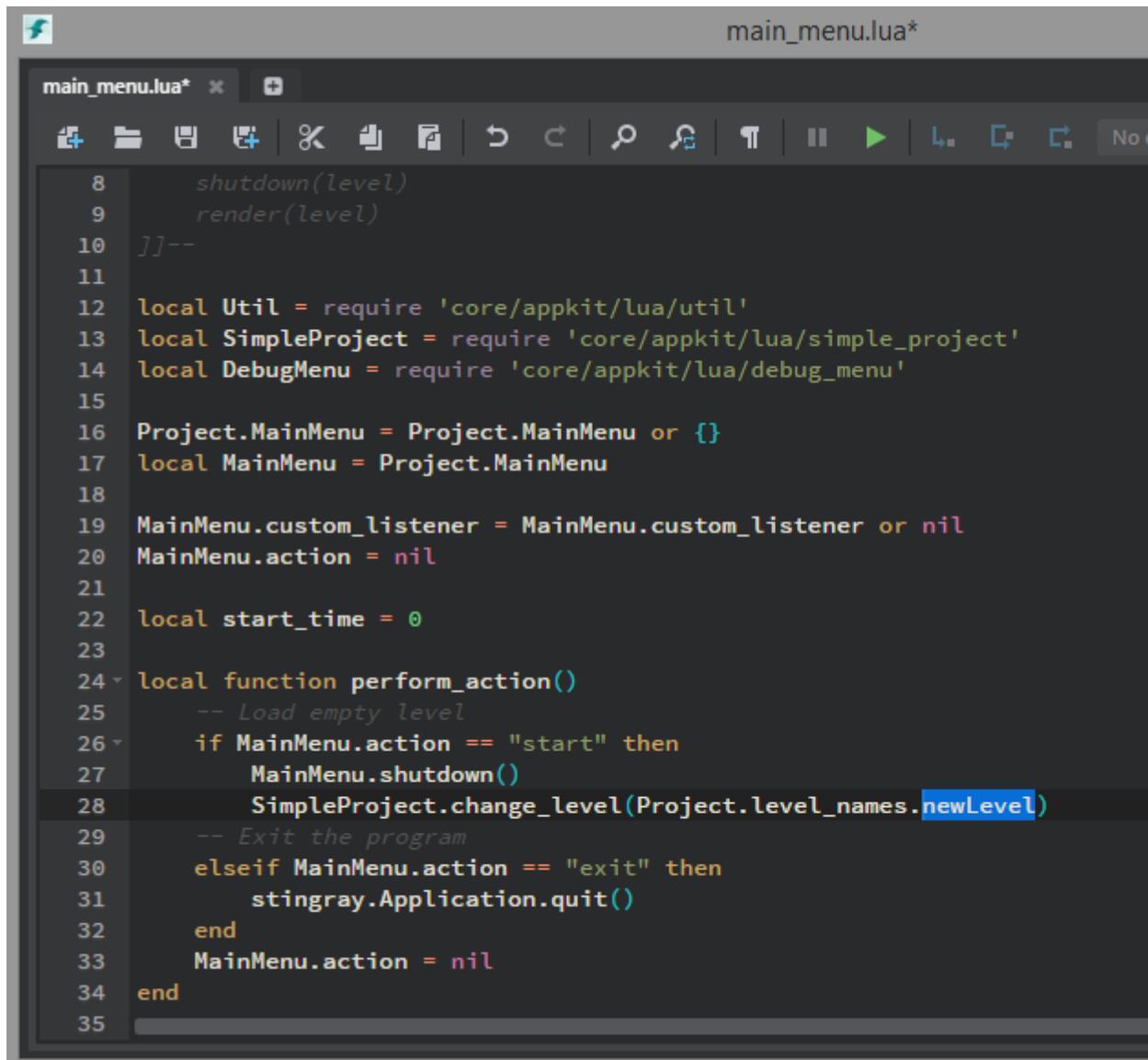
To change the level that is loaded when the user clicks on the “main menu” button you need to modify the following:

-Define the new level in the project.lua file



```
1 -----
2 -- This implementation uses the default SimpleProject and the Project
3 -- used to extend the SimpleProject behavior.
4
5 -- This is the global table name used by Appkit Basic project to extend
6 Project = Project or {}
7
8 require 'script/lua/flow_callbacks'
9
10 Project.level_names = {
11     menu = "content/levels/main_menu",
12     basic = "content/levels/basic",
13     newLevel = "content/levels/newLevel"
14 }
15
16 -- Can provide a config for the basic project, or it will use a default
17 local SimpleProject = require 'core/appkit/lua/simple_project'
18 SimpleProject.config = {
19     standalone_init_level_name = Project.level_names.menu,
20     camera_unit = "core/appkit/units/camera/camera",
21     camera_index = 1,
22     shading_environment = nil, -- Will override levels that have env se
23     create_free_cam_player = false, -- Project will provide its own pla
24     exit_standalone_with_esc_key = false
25 }
26
27 -- This optional function is called by SimpleProject after level, world
28
```

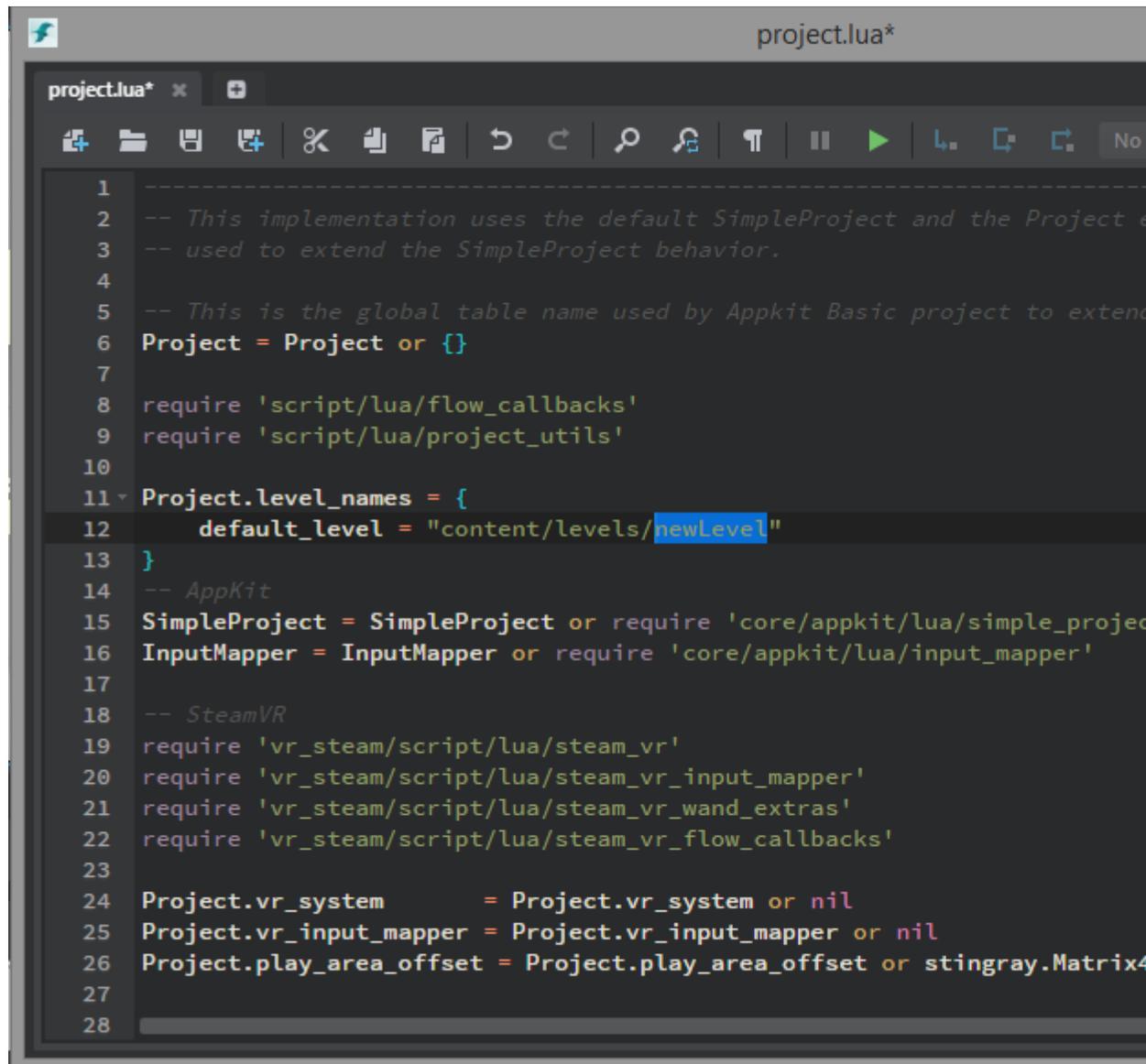
-Set the desired level as the action on the “Main Menu” button in the “main\_menu.lua” file:



```
8     shutdown(level)
9     render(level)
10 ]]--
11
12 local Util = require 'core/appkit/lua/util'
13 local SimpleProject = require 'core/appkit/lua/simple_project'
14 local DebugMenu = require 'core/appkit/lua/debug_menu'
15
16 Project.MainMenu = Project.MainMenu or {}
17 local MainMenu = Project.MainMenu
18
19 MainMenu.custom_listener = MainMenu.custom_listener or nil
20 MainMenu.action = nil
21
22 local start_time = 0
23
24 local function perform_action()
25     -- Load empty level
26     if MainMenu.action == "start" then
27         MainMenu.shutdown()
28         SimpleProject.change_level(Project.level_names.newLevel)
29     -- Exit the program
30     elseif MainMenu.action == "exit" then
31         stingray.Application.quit()
32     end
33     MainMenu.action = nil
34 end
35
```

## VR template

To change the default level loaded in the VR template you need to modify the following in the "project.lua" file:



```
1 -----
2 -- This implementation uses the default SimpleProject and the Project
3 -- used to extend the SimpleProject behavior.
4
5 -- This is the global table name used by Appkit Basic project to extend
6 Project = Project or {}
7
8 require 'script/lua/flow_callbacks'
9 require 'script/lua/project_utils'
10
11 Project.level_names = {
12     default_level = "content/levels/newLevel"
13 }
14 -- AppKit
15 SimpleProject = SimpleProject or require 'core/appkit/lua/simple_project'
16 InputMapper = InputMapper or require 'core/appkit/lua/input_mapper'
17
18 -- SteamVR
19 require 'vr_steam/script/lua/steam_vr'
20 require 'vr_steam/script/lua/steam_vr_input_mapper'
21 require 'vr_steam/script/lua/steam_vr_wand_extras'
22 require 'vr_steam/script/lua/steam_vr_flow_callbacks'
23
24 Project.vr_system = Project.vr_system or nil
25 Project.vr_input_mapper = Project.vr_input_mapper or nil
26 Project.play_area_offset = Project.play_area_offset or stingray.Matrix4
27
28
```

## First steps

Once an appropriate template is chosen for your project you would at this time export your assets from 3ds Max to the project folder of Stingray. These assets will not appear automatically in the viewport. You must place them manually.

\* If the assets were positioned in the desired place in 3ds Max you just have to place each asset in the viewport and then ensure that it is placed at coordinate 0, 0, 0

If needed, the next step would be to add lights and create new materials if needed

## Adding that extra eye candy

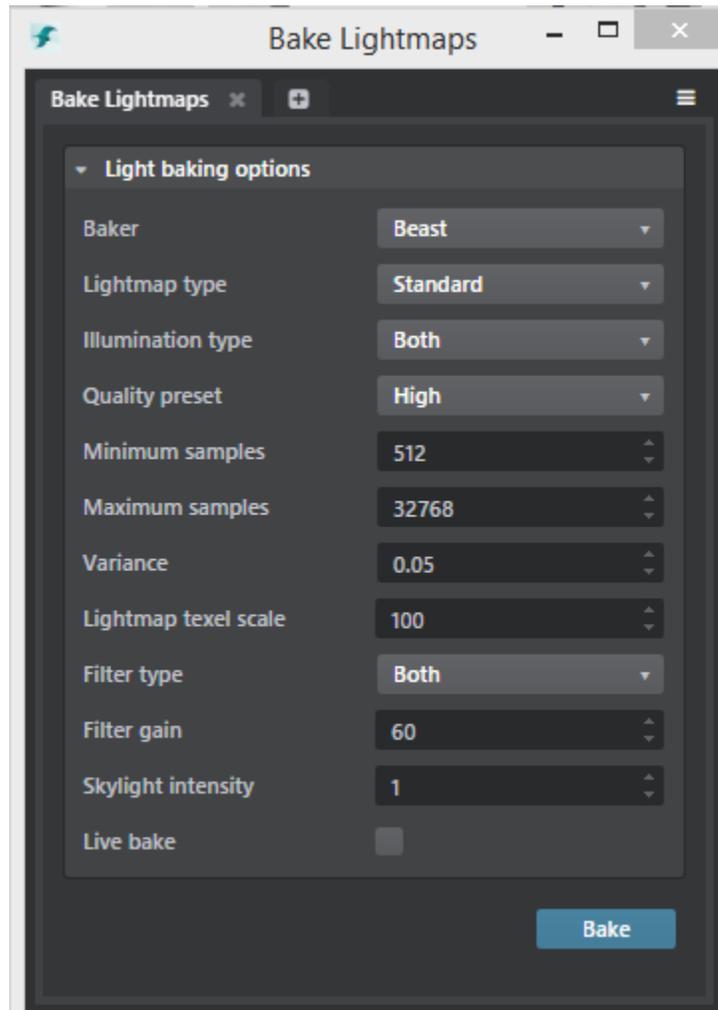
Since game engines are required to calculate and update what the viewer sees ideally more than 60 times per second (90 for VR) many complex and expensive features cannot be used real-time (i.e raytraced reflections, indirect lighting)

Techniques have been created to give effects similar to those expensive features but with a minimal performance hit

## Light Baking

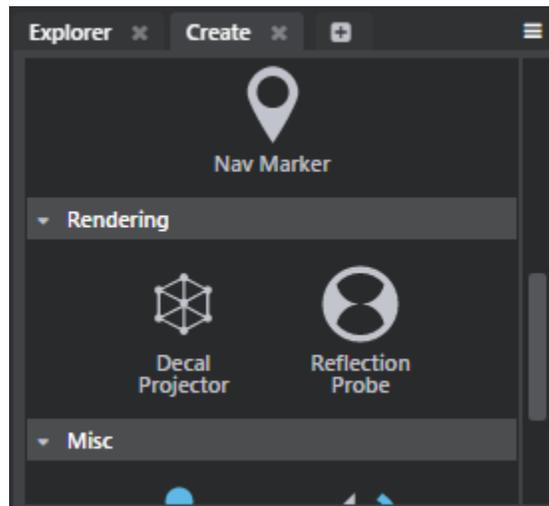
Light baking is the process of performing complex lighting calculations and then creating textures based on the results. Those textures are then applied to the objects in the scene to give the illusion of complex lighting. This is done on a per light basis and is not suitable in areas where objects or lights will change during project viewing (ex: moving a chair, turning off a light)

Window > Lighting > Bake Lightmaps



## Reflection Probes

Another expensive process are ray traced reflections. To give the illusion of those type of reflections we use reflection probes. These are helpers that are place centrally in a chosen area and then a 360 degree image is create from its point of view. This image is then used to fake reflections on objects placed in that area.



- Reflection Probes can be found in the Create panel under “Helpers”
- Once placed, cube maps can be generated by going into
  - Windows > Lighting & Shading > Lighting > Bake Reflection Probes

## Collisions

Objects by default will not interact with the player; this is done for performance reasons. At the bare minimum you need to define floors that the player can “collide” with so that they do not instantly fall indefinitely when the project begins. These areas also define where you can teleport to in when using the HTC Vive VR headset

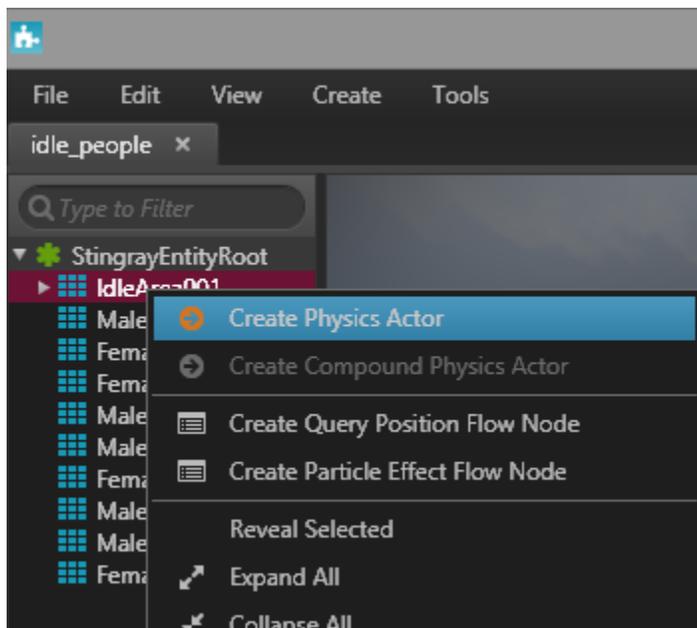
This is done by defining Physics Actors for the needed objects

For non-VR projects

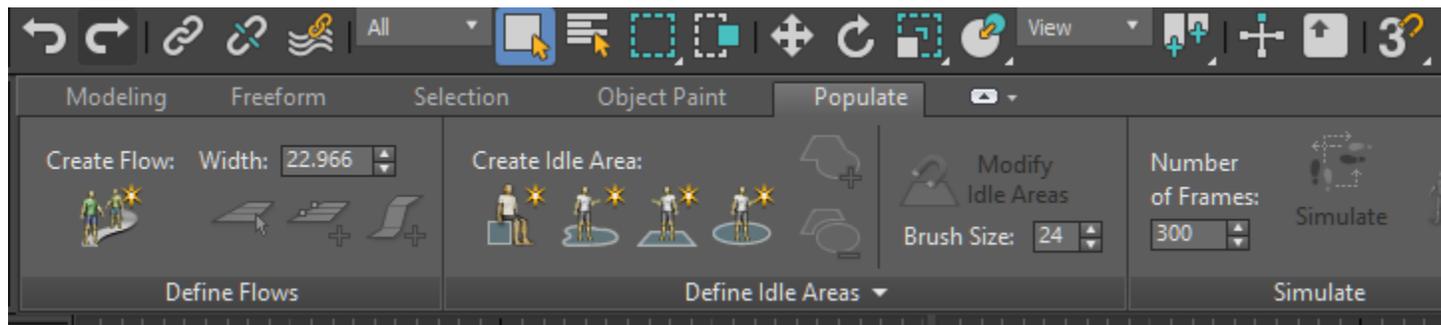
- Must be done on all floor objects that the user can walk on
- Objects that the user should not be able to walk though (ex: walls, furniture, railing, etc.)

For VR projects

- User cannot be physically restrained from going though objects so any floors that the user are allowed to teleport to should have physics actors created for them



## Populate, an alternative to RPC people



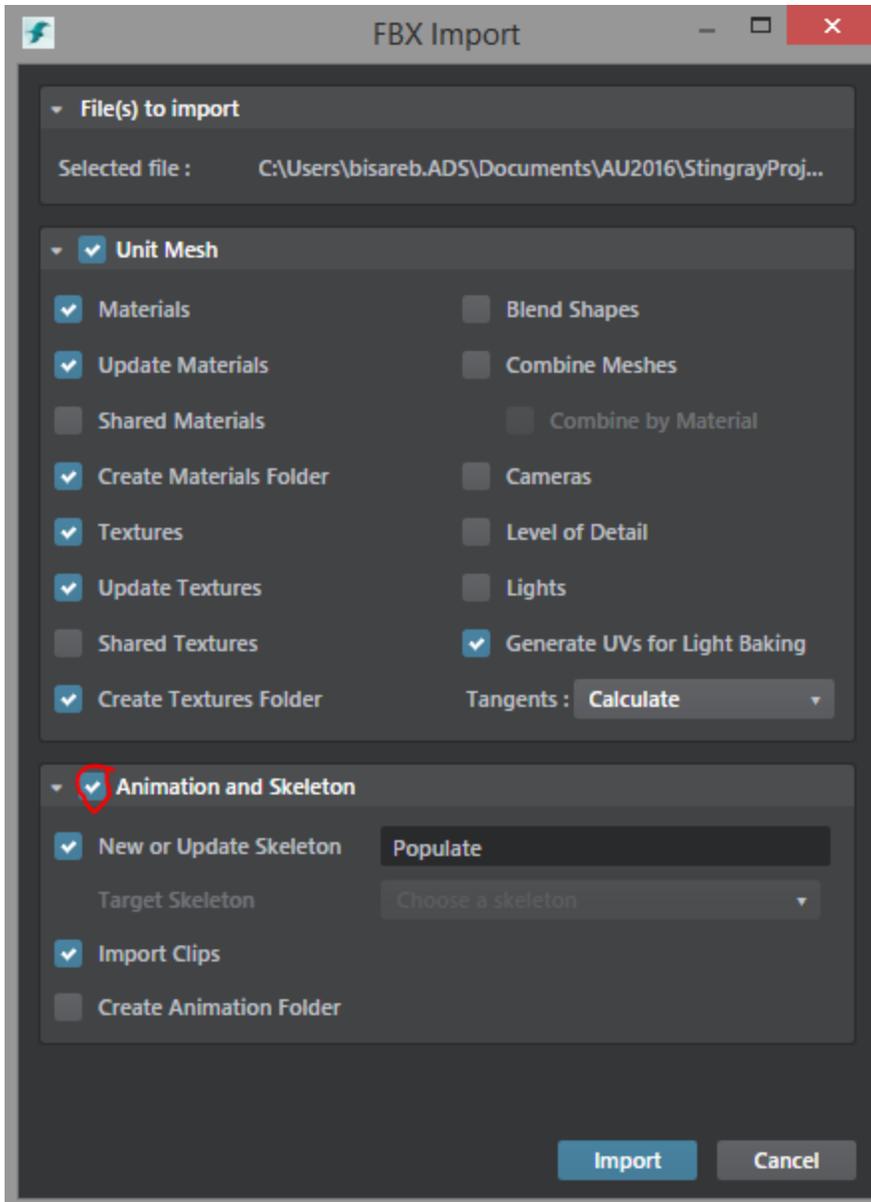
Populate is a simple tool to allow you to create animated people that can be used in your Stingray project.

The first step is to create either people walking along a path (flow) or standing in place (idle). You can use the tools within populate to subtly change the look and gender of the people.

Next step is to select the desired people (flow object not necessary) and bake the animation

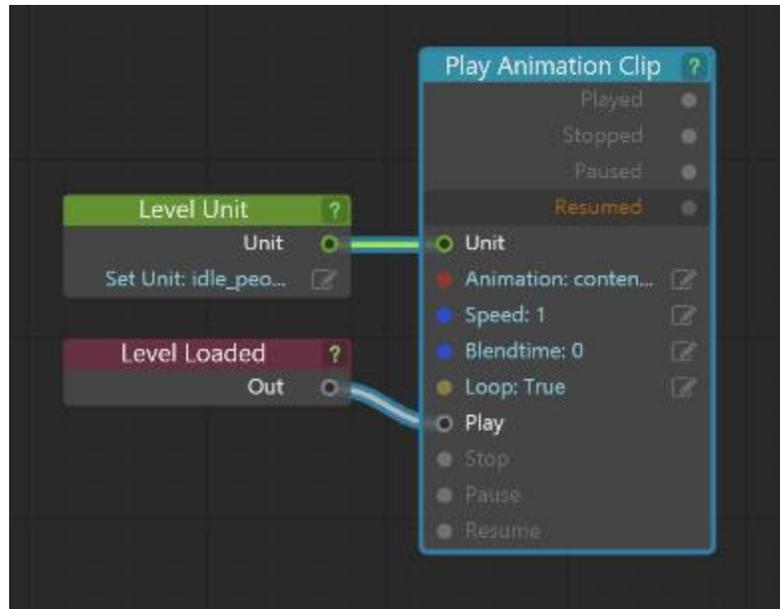
Populate tab > Edit Selection > Bake Selected

Once done select the people and associated bones and send them to Stingray. In Stingray ensure that you enable the Animation option



Next create the following Level Flow

- Event > Level Loaded
- Animation > Play Animation Clip
  - Animation: Choose the animation clip created from the Populate import
  - Connect the “out” from the Level Loaded node to the “Play” input of the “Play Animation Clip” node
- Data > Level Unit
  - Set Unit: Choose the Populate unit
  - Connect the “Unit” output of the Level Unit node to the “Unit” input of the Play Animation Clip

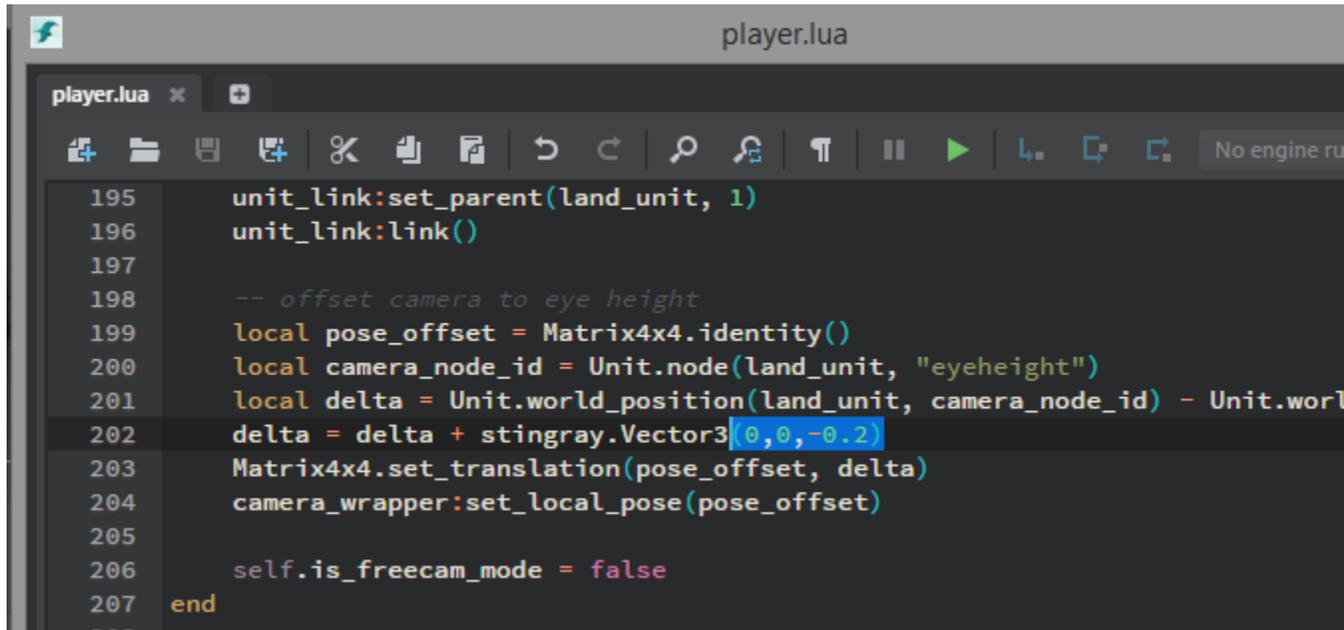


## Starting position and eye height

Changing the player start position can be done by changing the highlighted coordinates in the "project" lua file

```
project.lua  
  
22 create_free_cam_player = false, -- Project will provide its own player.  
23 exit_standalone_with_esc_key = false  
24 }  
25  
26 -- This optional function is called by SimpleProject after level, world and p  
27 -- but before lua trigger level loaded node is activated.  
28 function Project.on_level_load_pre_flow()  
29     -- Spawn the player for all non-menu levels. level_name will be nil if th  
30     -- is an unsaved Test Level.  
31     local level_name = SimpleProject.level_name  
32     if level_name == nil or level_name ~= Project.level_names.menu then  
33         local view_position = Appkit.get_editor_view_position() or stingray.V  
34         local view_rotation = Appkit.get_editor_view_rotation() or stingray.Q  
35         local Player = require 'script/lua/player'  
36         Player.spawn_player(SimpleProject.level, view_position, view_rotation  
37     elseif level_name == Project.level_names.menu then  
38         local MainMenu = require 'script/lua/main_menu'
```

Changing the eye height can be done by changing the highlighted coordinates in the "player" lua file



```
195     unit_link:set_parent(land_unit, 1)
196     unit_link:link()
197
198     -- offset camera to eye height
199     local pose_offset = Matrix4x4.identity()
200     local camera_node_id = Unit.node(land_unit, "eyeheight")
201     local delta = Unit.world_position(land_unit, camera_node_id) - Unit.world_position(land_unit, "eyeheight")
202     delta = delta + stingray.Vector3(0,0,-0.2)
203     Matrix4x4.set_translation(pose_offset, delta)
204     camera_wrapper:set_local_pose(pose_offset)
205
206     self.is_freecam_mode = false
207 end
```

## Creating a stand-alone package

### Windows

[http://help.autodesk.com/view/Stingray/ENU/?guid=stingray\\_help\\_deploying\\_package\\_for\\_windows\\_html](http://help.autodesk.com/view/Stingray/ENU/?guid=stingray_help_deploying_package_for_windows_html)

### Android

For Android you must first install the prerequisites found in the Stingray documentation

[http://help.autodesk.com/view/Stingray/ENU/?guid=stingray\\_help\\_deploying\\_platform\\_support\\_reqs\\_android\\_html](http://help.autodesk.com/view/Stingray/ENU/?guid=stingray_help_deploying_platform_support_reqs_android_html)

Then deploy the project using the steps found here

[http://help.autodesk.com/view/Stingray/ENU/?guid=stingray\\_help\\_deploying\\_package\\_for\\_android\\_html](http://help.autodesk.com/view/Stingray/ENU/?guid=stingray_help_deploying_package_for_android_html)

### iOS

[https://www.youtube.com/watch?v=0f6Zd\\_pep8Q](https://www.youtube.com/watch?v=0f6Zd_pep8Q)

