DEPLORABLE MOUNTAINEER

MOOREFIELD, WV

HTTPS://DEPLORABLEMOUNTAIN.WIXSITE.COM/GAMES

DEPLORABLE.MOUNTAINEER@GMAIL.COM

CLASSIC FPS GAME PROJECT DESIGN DOCUMENT

BY TODD D. VANCE

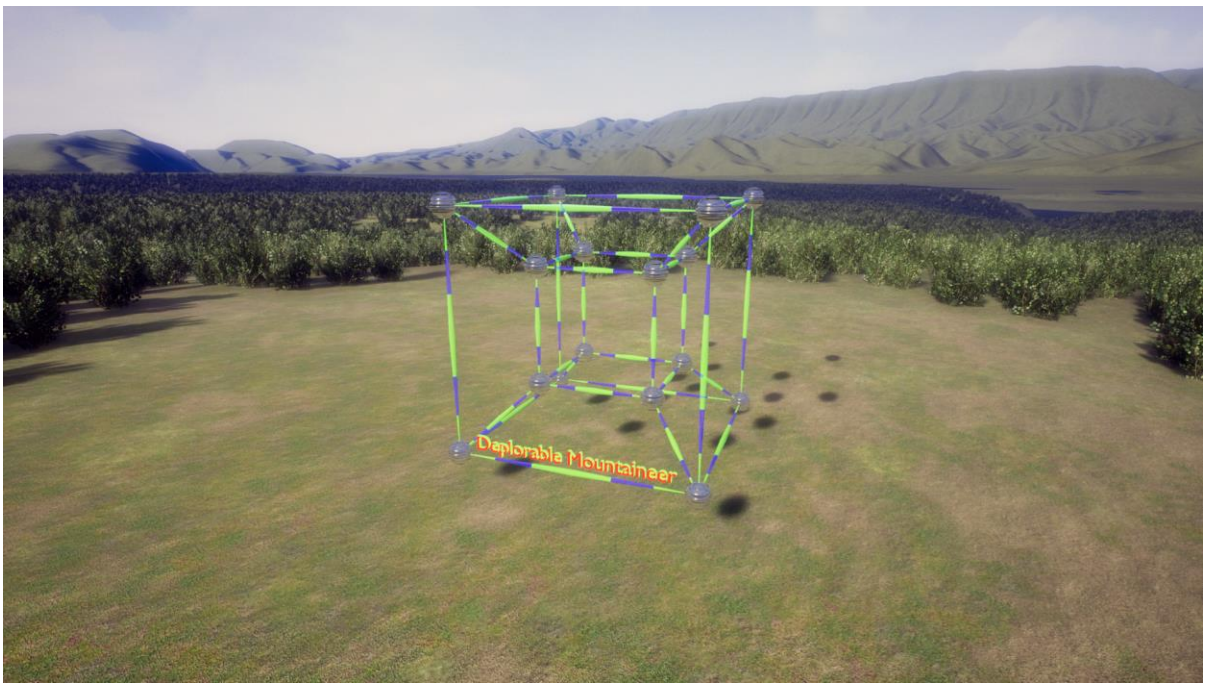


FIGURE 1: A MOCKUP OF THE SOUTH BRANCH VALLEY SURROUNDING MOOREFIELD, WV, WITH THE HYPERCUBE LOGO OF DEPLORABLE MOUNTAINEER SITTING ON THE GROUND.

FOREWORD

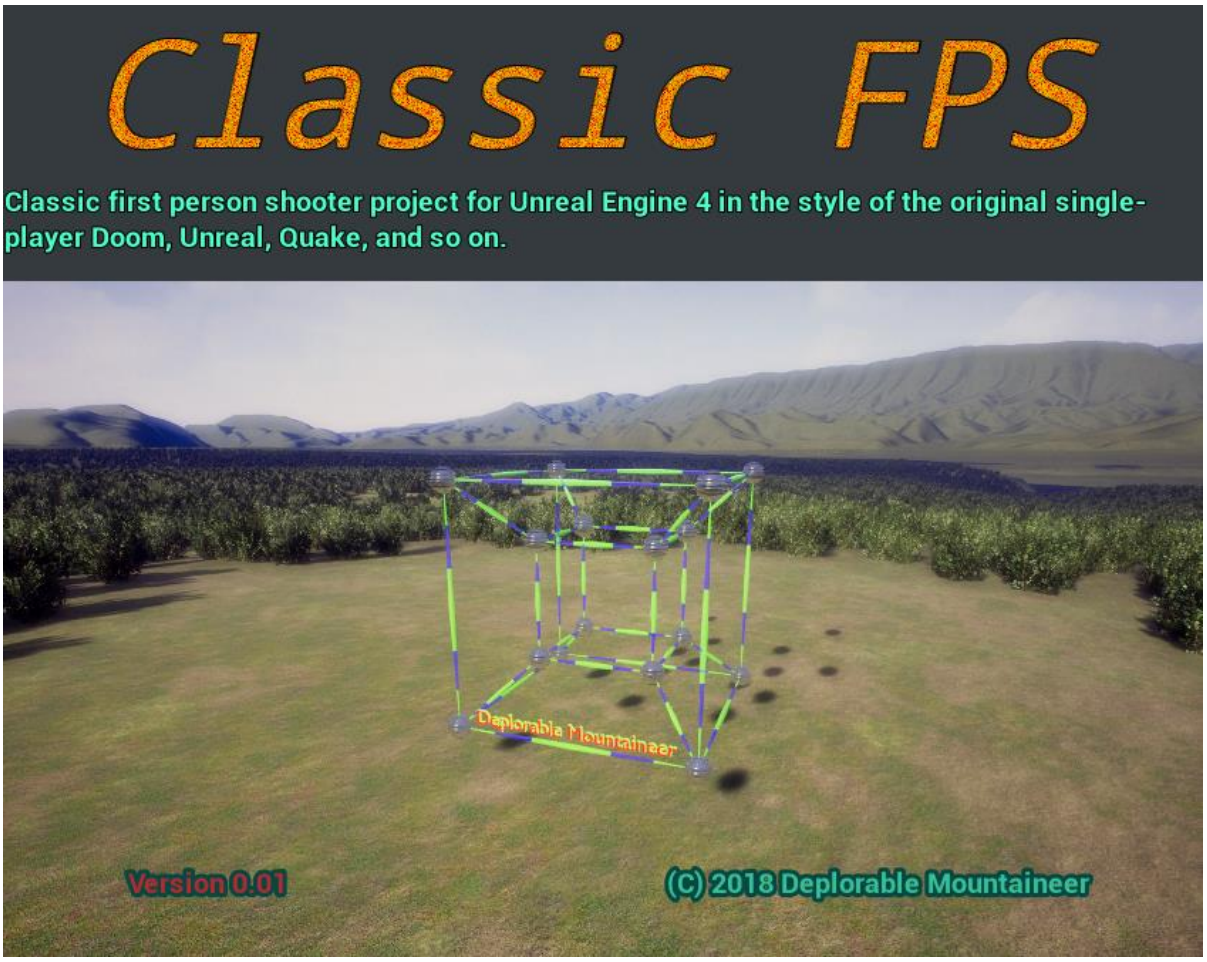
CONTENTS

TODO

- Make rotating actors controllable
- Light Effects: blink, color change, pulsate, flicker
- Delay Switch, switch logic
- Spawners
- Earthquake
- Conveyor belt
- Water current
- Rising/lowering water
- Water waves
- Underwater Volume
- Force fields
- Mirrors, including two way
- Teleporters
- End of Level
- Exploding crates and barrels
- Destructible meshes
- Turret and exploder (controllable)
- Pistons, crushing ceiling
- Sliding doors, vertical doors, cylinder doors, other doors
- Security cameras
- Crane?
- Secret Areas
- Thunder/lightning/rain
- Snow effects
- Escalator
- Debris spawning
- Ladder/climbable
- Fire Alarm
- Fire Extinguisher
- Fire Hydrant
- Landscape Materials
- Saw horse
- Ceiling Light Panels
- Rotary Switch
- Knife Switch
- False Ceiling
- Raised Floor
- Broken dingy Tiles
- Circuit Board
- Tube Lights
- Locker Door
- TV/Monitor Screen Images

- Computer equipment (desktop, laptop)
- Keypads and Keyboards
- Vent Louvres
- Rusted Metal Panels, Metal Panels
- Textured and wallpaper walls
- Vault Door, Freezer Door
- Office Chair
- Water Fountain
- Urinal
- Speaker
- Antennas (framework, dish, whip)
- Water Drops
- Bubbles
- Blackboard, Whiteboard
- File Cabinet, cupboards, drawers
- Glass Panels
- Wainscoting
- Books
- Bookshelf
- Curtains
- Bed, Mattress
- Tables
- Breaker Box, Junction Box, Gas Meter, Power Meter, etc.
- ATM
- Fountain
- Chain
- Rope
- Stoplight
- Barstool
- Bar
- Flashing Yellow Hazard Lights
- Hand Dryer, Towel Dispenser, Toilet Paper Dispenser
- Street Lamp
- Paintings
- Vending Machines
- Fish
- Elevator Panel/lights
- Soda Fountain
- Cellphone, Tablet, Ipod
- Flag
- Napkin Dispenser, Condiments, Booth benches
-

CHAPTER 1: GAME INFORMATION



Unique Project ID: {AED64A86-47D8-D36A-F0A9-2093602951A6}

Name: Classic FPS Version 0.01

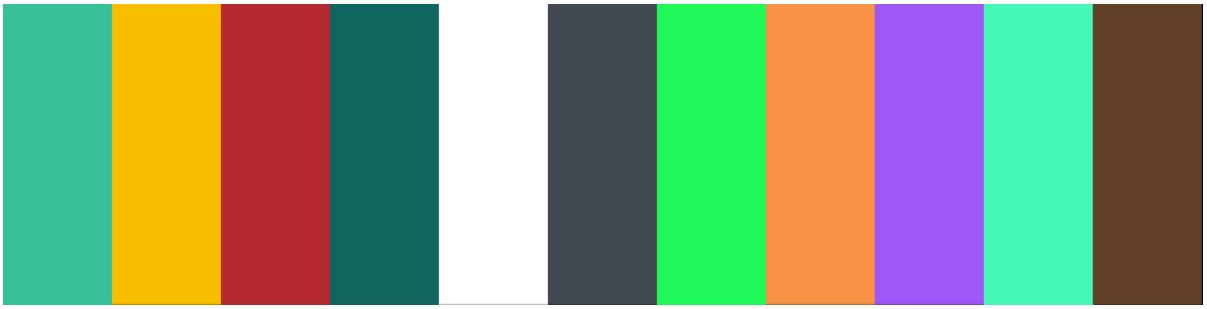
Description: Classic first person shooter project for Unreal Engine 4 in the style of the original single-player Doom, Unreal, Quake, and so on.

Company: Deplorable Mountaineer

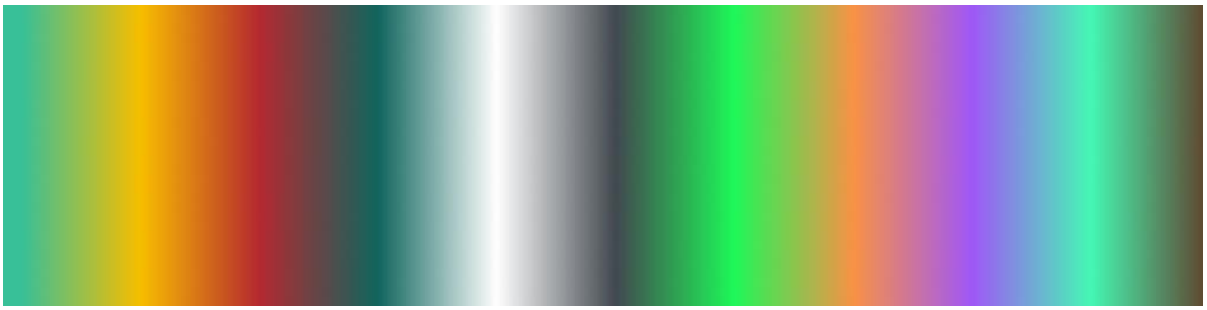
Website: https://deplorablemountain.wixsite.com/games

Contact: Deplorable.Mountaineer@gmail.com

Palette Colors: #38c098 (Shamrock), #F8BE00 (Amber), #B52930 (Tall Poppy), #10655E (Eden), #ffffff (White), #42484f (Mako), #1EF858 (Malachite), #F89245 (Tan Hide), #A057F8 (Heliotrope), #45F8B5 (Aquamarine), #613F29 (Irish Coffee)



Palette Gradient:



CHAPTER 2: FOLDER STRUCTURE

Under Content:

file://D:\\Documents\\Repos\\Deplorable_Mountaineer\\UE4\\ClassicFPS\\Content

CHAPTER 3: ASSETS

MATERIALS

PBR SETTINGS

For game use only--I made up some of the values.

file:///D:/Documents/Repos/Deplorable_Mountaineer/Spreadsheets/Materials.ods

MATERIAL MASTERS

Base

Most materials use this shader; has parameters for almost everything

Translucent_Expensive

A shader for translucent materials using the most expensive operations; use sparingly

Mask

A shader for RGB material masks

PARAMETER NAMES

file:///D:/Documents/Repos/Deplorable_Mountaineer/Spreadsheets/MaterialParameters.ods

MATERIAL FUNCTIONS AND LAYERS

MF_MaskLerpRGB

A function used for making complex RGB masked materials with smooth transitions

TEXTURES

Crosshairs

Cross 

X 

Dot 

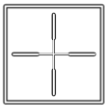
Triangle 

Trefoil 

Box 

Circle 

Rangefinder 

Box Cross 

Box Dot 

Box X 

Circle Cross 

Circle Dot 

Circle X 

Metal

dimpled, rough and scratchy, Rusty, Holey, verdigris, Brushed, galvanized, Corrugated, Grate, Fence, Louvred Vent, Duct

Rock

smooth, conglomerate, Sandy, Gravelly, Pitted, jagged, marble, slate, limestone, Granite, Sandstone

Icons

Ammo 

Up Arrow 

Carryable 

Controllable 

Controller 

Door 

Health 

Key 

Light 

Sky 

Pawn 

Ruler 

Shield 

Stairs 

Strength 

Switch 

Weapon 

Cloth

Fine Weave, Coarse Weave, Carpet

Decals

bullet holes, footprints, Stain, signs, puddles, Dirt, Scratches

Wall

Plaster, wood, wood with grain, particleboard, Wood with knots, brick, concrete, concrete block, Stone

Floor

tiles of various types and shapes, cobblestone, Concrete, Concrete grime, Concrete Panels, Concrete Tiles, Concrete grooved, Concrete Pitted, Concrete Plank-like

Ground

Dirt, Mud, Gravel, Grass, Snow, Patchy Grass, Ice, Dirty Ice, Rough Ice, Dirty Snow, Patchy Snow, Exposed Rock, muddy rocks, Moss, Dirt and Pebbles, Ground with Foliage

Utility

noise, clouds, gradients

Other

Painted metal, colored glass, patterned cloth, etc.

Leather, Rubber, Plastic, Ceramic, etc. to match PhysMats

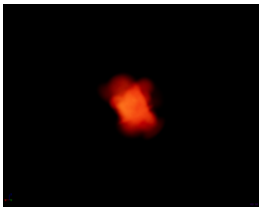
Hazard patterns

Grids and tile patterns, hex and square, etc. Holey patterns, Vent louvre patterns, etc.

Diamond Plate, Duct, Planks, Parquet

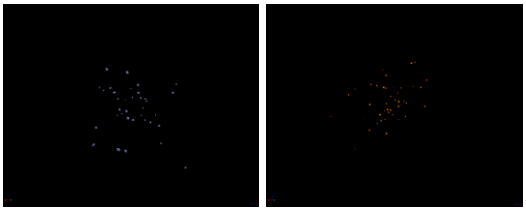
PARTICLES

ParticlesExplosions

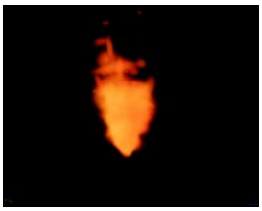


Hit effect

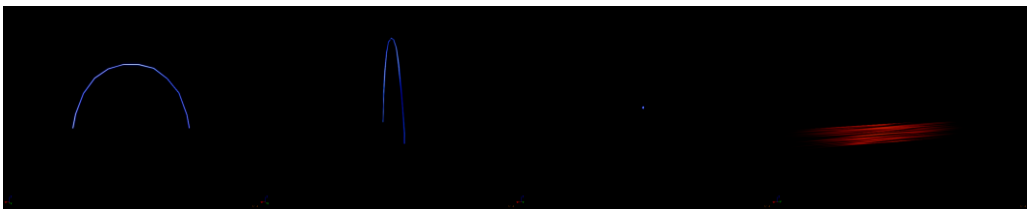
Sparks



Fires



Beams



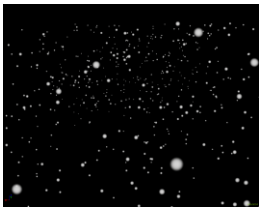
Projectile particles

Splash from water

Slime

Lava

Snow



Steam

Smoke

Lightning

Shrapnel

Gibs

Dust

Contrail

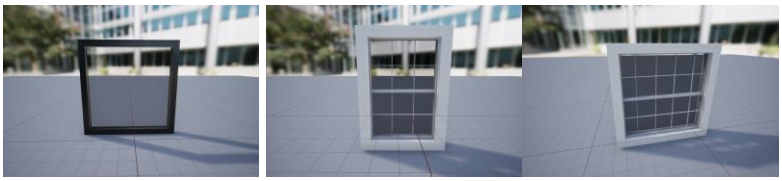
Light halo

Architectural Parameters

file:///D:/Documents/Repos/Deplorable_Mountaineer/Spreadsheets/Architectural.ods

Static Meshes

Window



Shapes

Cube, sphere, slope, Square pyramid, Tetrahedron, Octahedron, Icosahedron, Torus, Plane, Cube Chamfer, Cube Inset, Cone, Platform, Circular platform, Platforms of various polygon shapes, Annular platform, Arrows

Arch

Walls, Walls with window/door openings

Pipes and joints, fence parts,. Stair parts, hose parts

Props

Barrels, Crates, including exploding barrel, Boxes, bags

Bricks, Cement blocks, Rocks, Boulders

Cones, Saw horses, Signs, Manhole covers

Trash cans, chairs, desks, tables, Benches, Couches, Barstools,

Grates, vents,

Foliage

ACTOR BASE

PROPERTIES

Name

Unique Name (just use name-in-world)

Icon

Colors

Additional Tags (automatically populates list of actor tags)

STATE MACHINES

State machines are used for switches and for things controlled by switches. Thus a lockable door and a keycard reader for the door are both state machines, as is a light switch and the light itself.

A state machine is in a specific state. When asked to perform a transition, what actually happens depends on the state that it is in. Thus the state machine maps the current state and the requested transition to an action (possibly an animation), and the next state.

In addition, a state machine can be connected to other state machines in two ways. Forwarding transitions means whenever a transition occurs, a corresponding transition is made to occur on another state machine. The other way is sync'ing states. This means that whenever a state changes on one state machine, there is a corresponding state change on the other.

There are also two kinds of overrideable actions (events). The On Exec Transition action occurs when a transition occurs. The On Enter State action occurs when a new state is entered (including at initialization). Note that sometimes transitions do not cause a state change. In addition, sync'ing causes a state change with no transition.

DOOR STATES

For a door, we want to have lots of capabilities for maximum flexibility, and this means the state machine will be complex. The door in mind is a swinging door that can open either in or out, and can be locked, or locked from one direction. It can also be blocked open in or out. Finally, we allow for a physics mode, where the door just swings freely. Here is a table of all possible state transitions for our door.

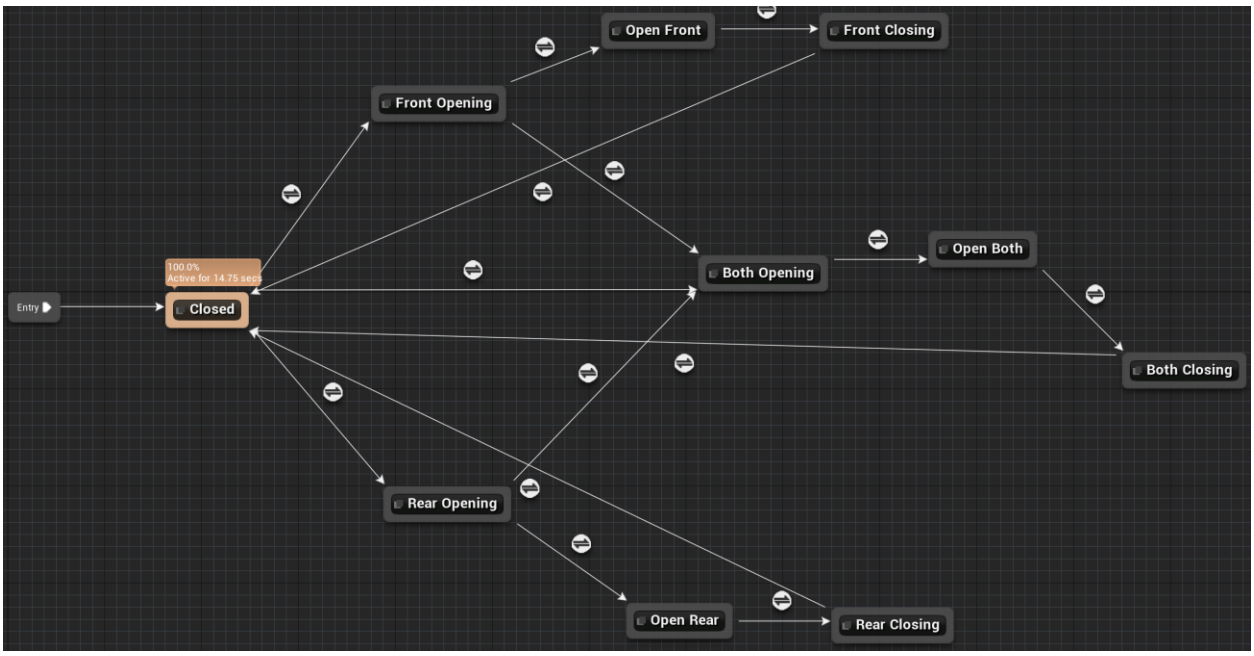
	State	
Start State	Closed	
Transitions	**From State**	**To State**
Open from Front	Open Forward	Closed
	Open Reversed	Open Reversed
	Closed	Open Reversed
	Open Forward and Locked Forward	Closed and Locked Forward
	Open Reversed and Locked Forward	Open Reversed and Locked Forward
	Closed and Locked Forward	Open Reversed and Locked Forward
	Open Forward and Locked Reverse	Closed and Locked Reverse
	Open Reversed and Locked Reverse	Open Reversed and Locked Reverse
	Closed and Locked Reverse	Closed and Locked Reverse
	Open Forward and Locked Forward and Locked Reverse	Closed and Locked Forward and Locked Reverse
	Open Reversed and Locked Forward and Locked Reverse	Open Reversed and Locked Forward and Locked Reverse
	Closed and Locked Forward and Locked Reverse	Closed and Locked Forward and Locked Reverse
	Blocked Open Forward	Blocked Open Forward
	Blocked Open Reversed	Blocked Open Reversed
	Physics Mode	Physics Mode
Open from Rear	Open Forward	Open Forward
	Open Reversed	Closed
	Closed	Open Forward
	Open Forward and Locked Forward	Open Forward and Locked Forward
	Open Reversed and Locked Forward	Closed and Locked Forward
	Closed and Locked Forward	Closed and Locked Forward
	Open Forward and Locked Reverse	Open Forward and Locked Reverse
	Open Reversed and Locked Reverse	Closed and Locked Reverse
	Closed and Locked Reverse	Open Forward and Locked Reverse
	Open Forward and Locked Forward and Locked Reverse	Open Forward and Locked Forward and Locked Reverse
	Open Reversed and Locked Forward and Locked Reverse	Closed and Locked Forward and Locked Reverse
	Closed and Locked Forward and Locked Reverse	Closed and Locked Forward and Locked Reverse
	Blocked Open Forward	Blocked Open Forward
	Blocked Open Reversed	Blocked Open Reversed
	Physics Mode	Physics Mode
Operate from Front	Open Forward	Closed
	Open Reversed	Closed
	Closed	Open Reversed
	Open Forward and Locked Forward	Closed and Locked Forward
	Open Reversed and Locked Forward	Closed and Locked Forward
	Closed and Locked Forward	Open Reversed and Locked Forward
	Open Forward and Locked Reverse	Closed and Locked Reverse
	Open Reversed and Locked Reverse	Closed and Locked Reverse
	Closed and Locked Reverse	Closed and Locked Reverse
	Open Forward and Locked Forward and Locked Reverse	Closed and Locked Forward and Locked Reverse
	Open Reversed and Locked Forward and Locked Reverse	Closed and Locked Forward and Locked Reverse
	Closed and Locked Forward and Locked Reverse	Closed and Locked Forward and Locked Reverse
	Blocked Open Forward	Blocked Open Forward
	Blocked Open Reversed	Blocked Open Reversed
	Physics Mode	Physics Mode
Operate from Rear	Open Forward	Closed
	Open Reversed	Closed
	Closed	Open Forward
	Open Forward and Locked Forward	Closed and Locked Forward
	Open Reversed and Locked Forward	Closed and Locked Forward
	Closed and Locked Forward	Closed and Locked Forward
	Open Forward and Locked Reverse	Closed and Locked Reverse
	Open Reversed and Locked Reverse	Closed and Locked Reverse
	Closed and Locked Reverse	Open Forward and Locked Reverse
	Open Forward and Locked Forward and Locked Reverse	Closed and Locked Forward and Locked Reverse
	Open Reversed and Locked Forward and Locked Reverse	Closed and Locked Forward and Locked Reverse
	Closed and Locked Forward and Locked Reverse	Closed and Locked Forward and Locked Reverse
	Blocked Open Forward	Blocked Open Forward
	Blocked Open Reversed	Blocked Open Reversed
	Physics Mode	Physics Mode
Close	Open Forward	Closed
	Open Reversed	Closed
	Closed	Closed
	Open Forward and Locked Forward	Closed and Locked Forward
	Open Reversed and Locked Forward	Closed and Locked Forward
	Closed and Locked Forward	Closed and Locked Forward
	Open Forward and Locked Reverse	Closed and Locked Reverse
	Open Reversed and Locked Reverse	Closed and Locked Reverse
	Closed and Locked Reverse	Closed and Locked Reverse
	Open Forward and Locked Forward and Locked Reverse	Closed and Locked Forward and Locked Reverse
	Open Reversed and Locked Forward and Locked Reverse	Closed and Locked Forward and Locked Reverse
	Closed and Locked Forward and Locked Reverse	Closed and Locked Forward and Locked Reverse
	Blocked Open Forward	Blocked Open Forward
	Blocked Open Reversed	Blocked Open Reversed
	Physics Mode	Physics Mode
Lock Forward	Open Forward	Open Forward and Locked Forward
	Open Reversed	Open Reversed and Locked Forward
	Closed	Closed and Locked Forward
	Open Forward and Locked Forward	Open Forward and Locked Forward
	Open Reversed and Locked Forward	Open Reversed and Locked Forward
	Closed and Locked Forward	Closed and Locked Forward
	Open Forward and Locked Reverse	Open Forward and Locked Forward and Locked Reverse
	Open Reversed and Locked Reverse	Open Reversed and Locked Forward and Locked Reverse
	Closed and Locked Reverse	Closed and Locked Forward and Locked Reverse
	Open Forward and Locked Forward and Locked Reverse	Open Forward and Locked Forward and Locked Reverse
	Open Reversed and Locked Forward and Locked Reverse	Open Reversed and Locked Forward and Locked Reverse
	Closed and Locked Forward and Locked Reverse	Closed and Locked Forward and Locked Reverse
	Blocked Open Forward	Blocked Open Forward
	Blocked Open Reversed	Blocked Open Reversed
	Physics Mode	Physics Mode
Lock Reverse	Open Forward	Open Forward and Locked Reverse
	Open Reversed	Open Reversed and Locked Reverse
	Closed	Closed and Locked Reverse
	Open Forward and Locked Forward	Open Forward and Locked Forward and Locked Reverse
	Open Reversed and Locked Forward	Open Reversed and Locked Forward and Locked Reverse
	Closed and Locked Forward	Closed and Locked Forward and Locked Reverse
	Open Forward and Locked Reverse	Open Forward and Locked Reverse
	Open Reversed and Locked Reverse	Open Reversed and Locked Reverse
	Closed and Locked Reverse	Closed and Locked Reverse
	Open Forward and Locked Forward and Locked Reverse	Open Forward and Locked Forward and Locked Reverse
	Open Reversed and Locked Forward and Locked Reverse	Open Reversed and Locked Forward and Locked Reverse
	Closed and Locked Forward and Locked Reverse	Closed and Locked Forward and Locked Reverse
	Blocked Open Forward	Blocked Open Forward
	Blocked Open Reversed	Blocked Open Reversed
	Physics Mode	Physics Mode
Unlock	Open Forward	Open Forward
	Open Reversed	Open Reversed
	Closed	Closed
	Open Forward and Locked Forward	Open Forward
	Open Reversed and Locked Forward	Open Reversed
	Closed and Locked Forward	Closed
	Open Forward and Locked Reverse	Open Forward
	Open Reversed and Locked Reverse	Open Reversed
	Closed and Locked Reverse	Closed
	Open Forward and Locked Forward and Locked Reverse	Open Forward
	Open Reversed and Locked Forward and Locked Reverse	Open Reversed
	Closed and Locked Forward and Locked Reverse	Closed
	Blocked Open Forward	Blocked Open Forward
	Blocked Open Reversed	Blocked Open Reversed
	Physics Mode	Physics Mode
Block	Open Forward	Blocked Open Forward
	Open Reversed	Blocked Open Reversed
	Closed	Closed and Locked Forward and Locked Reverse
	Open Forward and Locked Forward	Blocked Open Forward
	Open Reversed and Locked Forward	Blocked Open Reversed
	Closed and Locked Forward	Closed and Locked Forward and Locked Reverse
	Open Forward and Locked Reverse	Blocked Open Forward
	Open Reversed and Locked Reverse	Blocked Open Reversed
	Closed and Locked Reverse	Closed and Locked Forward and Locked Reverse
	Open Forward and Locked Forward and Locked Reverse	Blocked Open Forward
	Open Reversed and Locked Forward and Locked Reverse	Blocked Open Reversed
	Closed and Locked Forward and Locked Reverse	Closed and Locked Forward and Locked Reverse
	Blocked Open Forward	Blocked Open Forward
	Blocked Open Reversed	Blocked Open Reversed
	Physics Mode	Physics Mode
Unblock	Open Forward	Open Forward
	Open Reversed	Open Reversed
	Closed	Closed
	Open Forward and Locked Forward	Open Forward and Locked Forward
	Open Reversed and Locked Forward	Open Reversed and Locked Forward
	Closed and Locked Forward	Closed and Locked Forward
	Open Forward and Locked Reverse	Open Forward and Locked Reverse
	Open Reversed and Locked Reverse	Open Reversed and Locked Reverse
	Closed and Locked Reverse	Closed and Locked Reverse
	Open Forward and Locked Forward and Locked Reverse	Open Forward and Locked Forward and Locked Reverse
	Open Reversed and Locked Forward and Locked Reverse	Open Reversed and Locked Forward and Locked Reverse
	Closed and Locked Forward and Locked Reverse	Closed and Locked Forward and Locked Reverse
	Blocked Open Forward	Open Forward
	Blocked Open Reversed	Open Reversed
	Physics Mode	Physics Mode
Physics Mode	Open Forward	Physics Mode
	Open Reversed	Physics Mode
	Closed	Physics Mode
	Open Forward and Locked Forward	Physics Mode
	Open Reversed and Locked Forward	Physics Mode
	Closed and Locked Forward	Physics Mode
	Open Forward and Locked Reverse	Physics Mode
	Open Reversed and Locked Reverse	Physics Mode
	Closed and Locked Reverse	Physics Mode
	Open Forward and Locked Forward and Locked Reverse	Physics Mode
	Open Reversed and Locked Forward and Locked Reverse	Physics Mode
	Closed and Locked Forward and Locked Reverse	Physics Mode
	Blocked Open Forward	Physics Mode
	Blocked Open Reversed	Physics Mode
	Physics Mode	Physics Mode

ELEVATOR STATES

An elevator (in this framework) is very complex and there are multiple state machines involved.

For maximum flexibility, an elevator car can have doors on both the front and the back. One, the other, or both front and back doors can be opened. The elevator prop in this framework has both front and back doors, but can use materials to make it appear to have just front doors and no back doors, for maximum flexibility.

First, the *ABP_Elevator* Anim Blueprint has a state machine for animating the elevator doors. The primary non-animating states are for: doors closed, front doors open, rear doors open, and both doors open. Between various pairs of non-animating states are animating "transitory" states for: front door opening, front door closing, rear door opening, rear door closing, both doors opening, and both doors closing. The elevator skeletal mesh has animations for each of these six cases. Here is a screenshot of the state machine:



The animation blueprint is controlled by two Boolean variables: Open Front, and Open Rear. For example, if the elevator is in the Closed state, setting Open Front to True cause the front door to open and, when that finishes, the elevator is in the Open Front state. Then setting it to false cause the door to close and, when done, revert to the Closed state. Setting both variables True opens both front and rear doors. There are transitions from Front Opening and Rear Opening to Both Opening in case there is a tick or two between setting one variable and the other. More than a tick or two and the animation will have a slight glitch, so it is best to set both at the same time.

The animation blueprint has a third Boolean variable, Doors Open, which can be used to check if any doors are open. It is automatically set to True when either front or rear doors have begun opening, and set to False with both front and rear doors have fully closed.

The elevator is controlled by an (ordinary actor) blueprint, *BP_Elevator*, a child actor class of BP_ElevatorStates which is responsible for the state machine information.

CONTROLLABLE OBJECTS

Controllable objects have actions (of type Name) that can be called by a switch (a switch is really a state machine). Doors and Lights are examples, but so is a switch: so one switch can affect the state of another, allowing three-way switches, combo lock switches, etc.

A controllable object inherits from the BPI_Controllable interface, which provides the method "Do Action", which takes an Action:Name input to select the action, as well as input parameters (which are used or ignored depending on the action) Float Input:Float, Vector Input:Vector, Rotator Input:Rotator, Color Input:Linear Color, Transform Input:Transform, and Name Input:Name. It returns Success:Bool, True if successful, and False if not (for example, parameter out of range or action of given name does not exist).

One way to implement this method is to switch on the Action name, each branch calling a function named the same as the action and taking exactly the arguments they need. Another way is to inherit from BP_Controllable, and set the Allowed Actions array to be the list of action names, and override Do Action By Number to switch on the integer index of the action and perform the actions using the variables Float Input, etc., for inputs.

DOORS

DOOR ACTIONS

Open Forward

Open Reversed

Close

Lock

Unlock

Operate from Front

Operate from Rear

Auto Open from Front

Auto Open from Rear

- If operated or auto-opened from -X direction:
  - if open, close
  - if open reverse, close reverse
  - if closed and unlocked and can be operated/auto-opened, open
- If operated from X direction
  - if open, close
  - if open reverse, close reverse
  - if closed and unlocked and reversible, open reverse
  - if closed and unlocked and not reversible, open (note sliding doors typically not reversible)

DOOR PROPERTIES

Auto Close Delay: if 0, never auto close, otherwise after delay of that many seconds

Open Speed: might not be supported on all doors

Close Speed: might not be supported on all doors

Auto Open: if character is in trigger volume, opens automatically

Auto Open Reverse: if character is in reverse trigger volume, opens automatically in reverse

Operatable: "operate" opens the door; otherwise need to find a switch somewhere.

reversible: if false, open reverse fails.

Locked: if true, open and open reverse fail; if physics mode and closed, door becomes immovable

open state, closed state, open reversed state

MOVERS

Movers can be moved or rotated via actions. A mover is also a switch, so it has states and optionally can control something.

MOVER ACTIONS

Move To: world location vector

Move By: world direction vector

Rotate To: world rotator

Rotate By: additive world rotator

Transform To: World transform

Transform By: relative transform

Set Speed: Float

In addition are actions to transition between states.

LIGHTS

LIGHT ACTIONS

Actions typically follow a curve (ease-in ease-out) though some lights might have other behavior (E.G. fluorescent light flicker).

On: turn on to previous dimmer amount

Off: turn off regardless of dimmer amount

Toggle: (usual action for Operate)

- if on, turn off regardless of dimmer amount.
- If off, turn on to previous dimmer amount.

Set Dimmer Amount: value from 0 to 1, min brightness to max brightness

Set Light Color

Set Light Direction

LIGHT PROPERTIES

Max Brightness

Min Brightness (for dimmer, nonzero for technical reasons, but can be so near 0 it is essentially off)

Direction (rotator): direction a light faces, if applicable

Color

Dimmer Amount (from 0 to 1)

SWITCHES

SWITCH ACTIONS

Toggle: Change to next state (states are ordered in an array by integer; next state is the next-indexed state)

Set State: set to one of the states available to the switch.

SWITCH PROPERTIES

State: usually on or off, though could be more; represented by a value of type "Name"

On Enter State: mapping from states to actions on a controlled object (door, light, etc.); action taken when state is entered.

On Exit State: mapping from states to actions on a controlled object (door, light, etc.); action taken when state is exited.

STATE MACHINE

The state machine class is a base class for lots of blueprint classes having complex logic. This includes switches, controllable lights, controllable or lockable doors, and so on.

A state machine has the following properties that can be set on each instance:

- States: A set of names of all states this state machine can be in.
- Current State: a name, and tells which state the machine is currently in. Setting this value on an actor placed in the scene will cause the actor to start in this state.
- Transitions: A map from transition names to STransitionMap structs. Each STransitionMap is a map from names of states (the from state) to names of states (the to state). When a transition of the given name is executed, if the current state is a from state in the transition map, the next state will be the to state in the transition map. If the current state is not a from state in the transition map, the next state will be the invalid state.

Forward Actions To: An array of BP_StateMachine actors. Whenever a transition of a given name is executed by this state machine, all state machines in this array will automatically execute a transition of the same name. This lets a state machine be a switch that controls another state machine (such as a light). ***Warning***: do not create loops among forwarded actions, or the game will not play. In particular, never forward to self. ***Warning***: watch out for race conditions – Do not have multiple state machines forward to the same state machine in the same tick or the result is unpredictable.

A state machine has the following public methods:

- Execute Transition(Transition Name:Name): New State – Execute the transition of the specified name, and return the new state. The new state will be invalid if the transition does not exist or if it does not apply to the current state.
- Try to Execute Transition(Transition Name:Name, Current State:Name, Next State:Name):Boolean – this should be overridden by all child blueprint classes. Overriding classes should call the parent method to ensure all behaviors associated with a given transition are executed. True is returned if some behavior is executed. Otherwise, False is returned, even if the transition is valid, indicating no further behavior was executed.

- Get Current State():Name – return the state the state machine is in.
- Is Current State Valid():Boolean – Return true if the state machine is in a valid state, false if not.

Note that when the state machine transitions to an invalid state for any reason, this fact is logged and printed when run in the editor, but not in a stand-alone game.

SINGLE-ACTIVATION SWITCH

A single-activation switch is like a push button, where pressing the button causes something to happen, but there is no state change. Thus, a single-activation switch has no state.

	State	
Start State	Default	
Transitions	**From State**	**To State**
Operate	Default	Default

TWO-POSITION SWITCH

A two-position switch is like a wall lightswitch: it is either on or off. The allowed actions are: turn on, turn off, and toggle (flip to the other state).

	State	
Start State	Off	
Transitions	**From State**	**To State**
Turn On	Off	On
	On	On
Turn Off	Off	Off
	On	Off
Toggle	Off	On
	On	

FOUR-POSITION SWITCH

A four-position switch is like the one controlling a lamp with a three-way bulb. The positions (states) are off, low, medium, and high. In addition to actions selecting a position, there are actions to advance and to go back.

	State	
Start State	Off	
Transitions	**From State**	**To State**
Turn Off	Off	Off
	Low	Off
	Medium	Off
	High	Off
Set to Low	Off	Low
	Low	Low
	Medium	Low
	High	Low
Set to Medium	Off	Medium
	Low	Medium
	Medium	Medium
	High	Medium
Set to High	Off	High
	Low	High
	Medium	High
	High	High
Next State	Off	Low
	Low	Medium
	Medium	High
	High	Off
Previous State	Off	High
	Low	Off
	Medium	Low
	High	Medium

GAMEPLAY BASE

Gameplay Base

IN-LEVEL STATE (CHECKPOINT)

The In-Level state refers to that which must be saved and restored to restart a game at a checkpoint. For brevity, we shall call this a Checkpoint henceforth.

Among the data to be saved is the state of all characters (player and NPCs) in the game, the state of all moveable items, the state of all state machine items (switches, lights, doors, etc. that can be changed by the player), the state of all pickups and pickup spawners in the level, and other things.

Character Checkpoint Constant State	Character Checkpoint Transient State	State Machine Checkpoint	Pickup Spawner Checkpoint	Moveable Object Checkpoint
Base Jump Velocity	Crouch Button Down	Current State	Ready to Pick Up	Transform
Max Walk Speed Crouched	Jump Button Down		Ghost Time	
Max Fly Speed	Third Person Camera Arm Length			
Max Swim Speed	Health			
Max Walk Speed Crouched	Strength			
Base Running Jump Velocity	Current Weapon Slot			
Min Velocity for Falling Counter	Third Person Mode			
Max Survivable Fall Time	Enemies Killed			
Normal Camera Boom Location	Secrets Found			
Crouch Camera Boom Location	Powerups Found			
Accuracy	Weapons Found			
Aggressiveness	Ammo Found			
Alertness	Health Found			
Jumpiness	Keys Found			
Map Awareness	Falling Counter			
Reaction Time	FOV			
Tactics	Carried Item			
First Gun	Inventory			
Starting Ammo	Weapon Inventory			
Weapon Toss Impulse	Shielding			
Weapon Priority	Temporarily Suspend Fall Damage			
Multiplier per Strength Increment	State			
Headshot Multiplier	Time of Last Transition			
Operate Range	Number of Times State Ticked since Entry			
Operate Impulse	Focused Waypoint			
Team	Current Waypoint			
Spring Arm Multiplier per Increment	Focused Ammo			
Min FOV	Focused Teammate			
Max FOV	Focused Health			
Num Zoom Levels	Focused Enemy			
Name	Time Enemy Detected			
Icon	Random Stream			
Colors	Location			
Additional Tags	Rotation			
Name to Override Data Table				
WantsHealthThreshold				
NeedsHealthThreshold				
WantsAmmoThreshold				
NeedsAmmoThreshold				
InMortalDangerThreshold				
SuspectEnemyTime				
TeammateAssistFlockDistance				
NearBotOrPickupOrWaypointDistance				
TimeTillBored				
FarFromWaypointsDistance				
TooCloseToAttackDistance				
Debug				
Waypoint Actors				
Collision Probe Distance				
Air Control				
Air Control Boost Multiplier				
Air Control Boost Velocity Threshold				
Brake Deceleration Falling				
Carry Distance				
Crouched Character Height				
Default Ground Friction				
Falling Lateral Friction				
Mass				
Max Step Height				
Min Time for Fall Damage				
Minimum Strength				
Strength Adjust				
Walkable Floor Angle				
Character Height				
Character Width				
Auto-Kill Fall Time				
Gravity				

In addition, the name of the level has to be saved.

GAMEPLAY STATES

Gameplay States

MAIN MENU

Main Menu

Available choices: New Game, Load Game, Options, Statistics, Quit

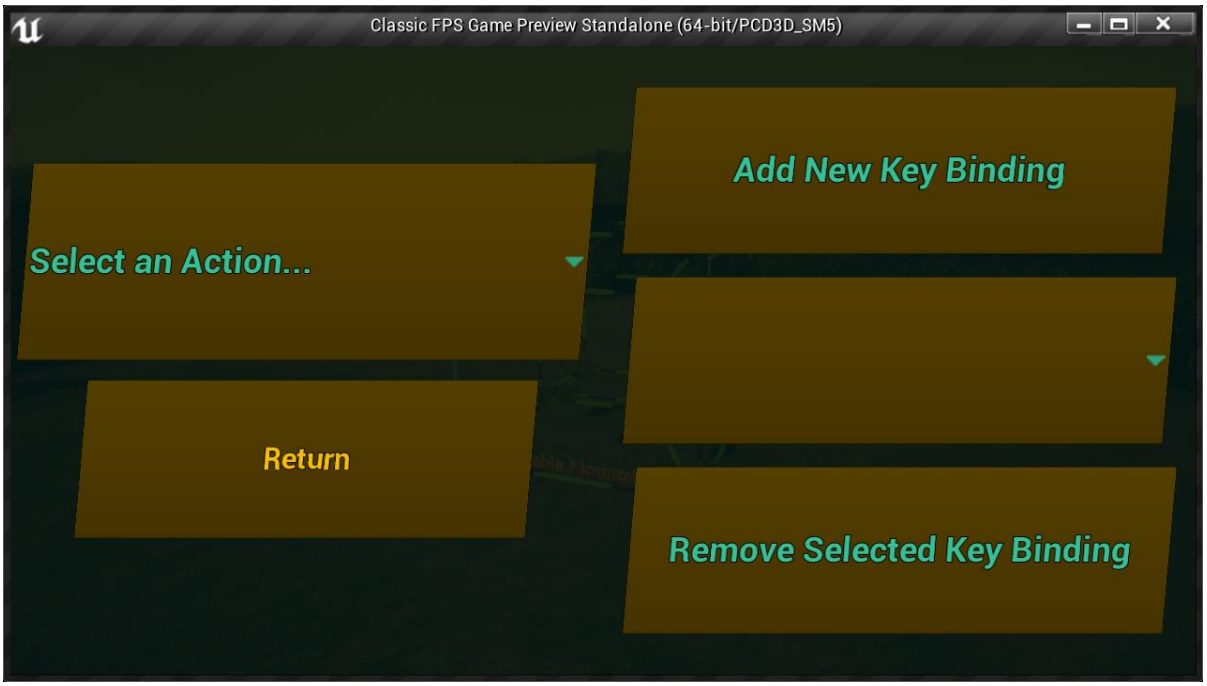
The main menu is brought up by the BeginPlay event of the Level Blueprint of the Entry map. This same level blueprint has properties UI Colors and UI Text for changing various menu properties for all menus.

New Game: Select Skill, Select Episode, Play, Return

Load Game: Select Game, Play, Return

Options: Action Input, Mouse and Axis Input, Graphics and Sound, Weapons, Return

Action Input: Add New Key Binding, Remove Selected Key Binding



Mouse and Axis Input: invert mouse, mouse sensitivity

Statistics: top scores, achievements

Graphics and Sound: Resolution, Effects Volume, Music Volume

Weapons: Priority, Auto Switch on Pickup (always, never, if better) Switch to Best Weapon when Out of Ammo

MAIN MENU CLASSES

BP_Pawn_Menu:

- Maintains Last Key Pressed, Last Key or Modifier Pressed, and Last Chord Pressed
- Chord to Text function
- Clear Keys event

MainMenu_GameMode:

- Get Bindings
- Action Key Mapping to Text
- Debug Print Bindings
- Axis Key Mapping to Text

BP_MainMenu (HUD class):

- Event Play Game
- Maintains array of First Levels of Episodes

- Sets input mode to UI and game, and shows mouse cursor for menu use

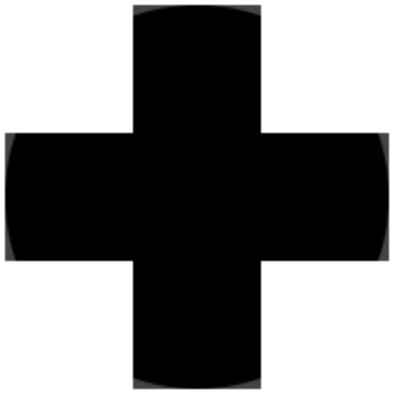
PAUSE MENU

HUD

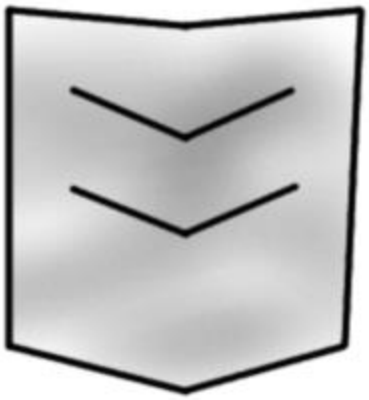
Ammo on bottom right, change color when low, with flash when it turns low and audible noise;

Radar Top right (as a powerup?), different color for enemies that are close; compass style

Health on bottom left; flash, audio warning when turns low, change color when low.



Next to health are Strength, Shield, and Keys.



Next to ammo are the powerups: Turbo, Night vision, Lift belt, and Radar.

T | N | L | R

Crosshair: fine lines, duplex (outer parts of lines highlighted); also dot and range-finding for long-range weapons.

ClassicFPS Game Mode

- Player is spawned at the beginning of the level, or at a saved point
- When player dies, can respawn at beginning or a saved point (state reset to the point)
- Upon reaching the end of level volume, there is a summary screen followed by starting the next level, keeping all inventory except keys.
- The Game Mode class maintains the list of weapons and default slots, up to 9 of which can be mapped to keys/buttons.

ClassicFPS Game State

- Current level
- Levels Completed
- Strength Achieved

ClassicFPS Player State

- Skill Level

- Name

Gameplay Effects

Hurt screen

Red tint, red dirt mask, extreme vignetting, extreme chromatic aberration; effect stronger as health decreases

Dead screen

Night Vision screen

Turbo screen

Low health screen?

Health increase screen

Shield used screen

Effects when enemy hit or killed

Pickup Effects

Low Ammo/Health/Inventory effects

In Battle Effects

Boredom Effects

Goal Achieved Effects

CHARACTER

DEFAULT CHARACTER PROPERTIES

file:///D:/Documents/Repos/Deplorable_Mountaineer/Spreadsheets/DefaultPlayerCharacter.ods

CHARACTER CAPABILITIES

Assuming for falling damage, falling on default surface, health at 100 and no shielding. Computed values are rounded. Max survivable fall height actually varies according to physmat of surface hit. For jump height, remember to add max step height to determine max height of platform that can be jumped onto. Also all values appear to be approximate: in-game values differ slightly from computed values for reasons the author doesn't know.

file:///D:/Documents/Repos/Deplorable_Mountaineer/Spreadsheets/CharacterCapabilitiesComputed.ods

Formulas used (assumes no air resistance, assumes character movement is physically accurate):

Jump Height: jump velocity * time + 0.5*-980*G*time*time

Velocity curve when jumping: Jump velocity – 980*G*time

Time for (single) jump: time = jump velocity / 980 / G

Double jumps double the height and time, but not velocity.

Action Bindings

file:///D:/Documents/Repos/Deplorable_Mountaineer/Spreadsheets/DefaultActionBindings.ods

Axis Bindings

file:///D:/Documents/Repos/Deplorable_Mountaineer/Spreadsheets/DefaultAxisBindings.ods

Character Properties

file:///D:/Documents/Repos/Deplorable_Mountaineer/Spreadsheets/DefaultCharacterProperties.ods

Character Property Value Modification

HEALTH

On picking up a health item, its value is added to the current health, capped at 100 (no cap if it is a super health item). If there is no change in health from the item, the item will not be picked up.

On taking damage, the health is diminished by the modified amount of damage, where the amount of damage is modified both by strength and amount of shielding possessed. Sometimes damage is multiplied if it hits certain bones instead of others (such as a head shot).

STRENGTH

The strength of most NPCs is fixed according to the character the NPC represents. The strength of the player can increase with "experience." The strength can temporarily increase upon consuming a turbo power boost ("Nitro") power up, but reverts when the power up runs out.

CURRENT WEAPON SLOT

This is the slot number of the weapon currently being carried. Weapons are numbered from 1 through unlimited (no hard limit to number of weapons, but only 1 through 9 can be selected by a keypress, others must be scrolled to with the scroll wheel, for example). It is changed when a weapon is dropped or selected. If dropped or tossed or stashed, the slot changes to 0 (no weapon). If selected (manually, or if auto-select is triggered in certain cases), it changes to whatever weapon slot is selected.

THIRD PERSON MODE

If true, the character is in third-person mode (relevant to player only).

ENEMIES KILLED

A list of unique names of enemies killed, for statistics purposes. Added to whenever this character kills someone, and reset on starting or restarting a level.

SECRETS FOUND

A list of names of secrets found, for statistics purposes. Secrets given the same name will be considered the same secret and not be counted more than once (thus allowing multiple entry points to a secret area). Added to whenever this character overlaps a "Secret Area" volume not yet found, and reset on starting or restarting a level.

POWERUPS FOUND

A list of unique names of powerups found, for statistics purposes. Added to whenever a character walks over or attempts to pick up a powerup, whether or not successful, and reset on starting or restarting a level.

WEAPONS FOUND

A list of unique names of weapons found, for statistics purposes. Added to whenever a character walks over or attempts to pick up a weapon, whether or not successful, and reset on starting or restarting a level.

AMMO FOUND

A list of unique names of ammo pickups found, for statistics purposes. Added to whenever a character walks over or attempts to pick up ammo, whether or not successful, and reset on starting or restarting a level.

HEALTH FOUND

A list of unique names of health pickups found, for statistics purposes. Added to whenever a character walks over or attempts to pick up health, whether or not successful, and reset on starting or restarting a level.

KEYS FOUND

A list of names of keys found, for statistics purposes, and to determine if character can pass through certain areas or not. Keys with the same name are considered equal. Added to whenever a character picks up a key not yet found, and reset on starting or restarting a level.

FALLING COUNTER

How long (in seconds) a character has been falling at more than a specified minimum velocity, used to compute damage from falling. It is incremented while the character is falling at greater than the minimum velocity, and reset when the player lands (whence damage is taken if appropriate).

FOV

Field of view in degrees, relevant only to the player. Adjusted when the player changes zoom level.

CARRIED ITEM

A struct holding data for the item currently being carried (so it can be reproduced on loading a saved game). It is modified when a player picks up or drops a physics item.

INVENTORY

An array of structs, each of which holds data for an inventory item being carried (so it can be reproduced on loading a saved game). It is modified when a player picks up or drops an inventory item of the type that stays with the character (such as a weapon).

THIRD PERSON CAMERA ARM LENGTH

Follow distance of the third-person camera. Relevant only for the player, and then only in third-person mode. Can be adjusted by the player.

CHARACTER STRUCTURE

Note that there is currently a bug (as of Version 4.18.3) in UE4 that causes the third-person character mesh to sometimes disappear in a level when the game isn't being played.

The first-person arms mesh is still visible, though. It can be fixed by recompiling the appropriate character blueprint. It appears to be harmless: the mesh reappears when you play the game.

A character has a BP_Character_Base as its parent, which in turn has the built-in Character class as its parent. Thus it already comes with a Capsule component as its root, as well as an arrow component (for convenience), and a skeletal mesh component. It also has a character movement component.

The assets in this project are designed for a character 192cm high and 84cm wide, so the capsule half-height is 96, and the capsule radius is 42. One can use a different-sized character, but things like doorway sizes might have to be adjusted.

The camera boom is attached to the capsule component, and the camera is attached to the camera boom. By shortening the camera boom to 0 length, we achieve first-person mode; otherwise, there is an adjustable-length boom third person mode.

The camera has a first-person arms mesh attached to it, visible only to the owning player and in first person mode (in which the other mesh, the third-person mesh, is made invisible to the owning player, but not to others).

The camera also has a box volume attached to it, which in turn has postprocessing components attached for various in-game effects.

Character Operating Items

When a character "operates" an item, it first checks if the item is an actor implementing the BPI_Operatable interface. If so, it calls the actor's Operate function.

If not, it checks if it is a pickup. If so, it tries to pick up the item. If not or if picking up fails, it checks if it simulates physics. If so, and if it is not too heavy, the character carries the item.

When the Operate action occurs and the character is already carrying something, it drops the item instead of performing the usual Operate procedure.

Text Messages

Characters can receive text messages. In a blueprint, on a Character actor, just call "Receive Text Message" with the appropriate inputs.

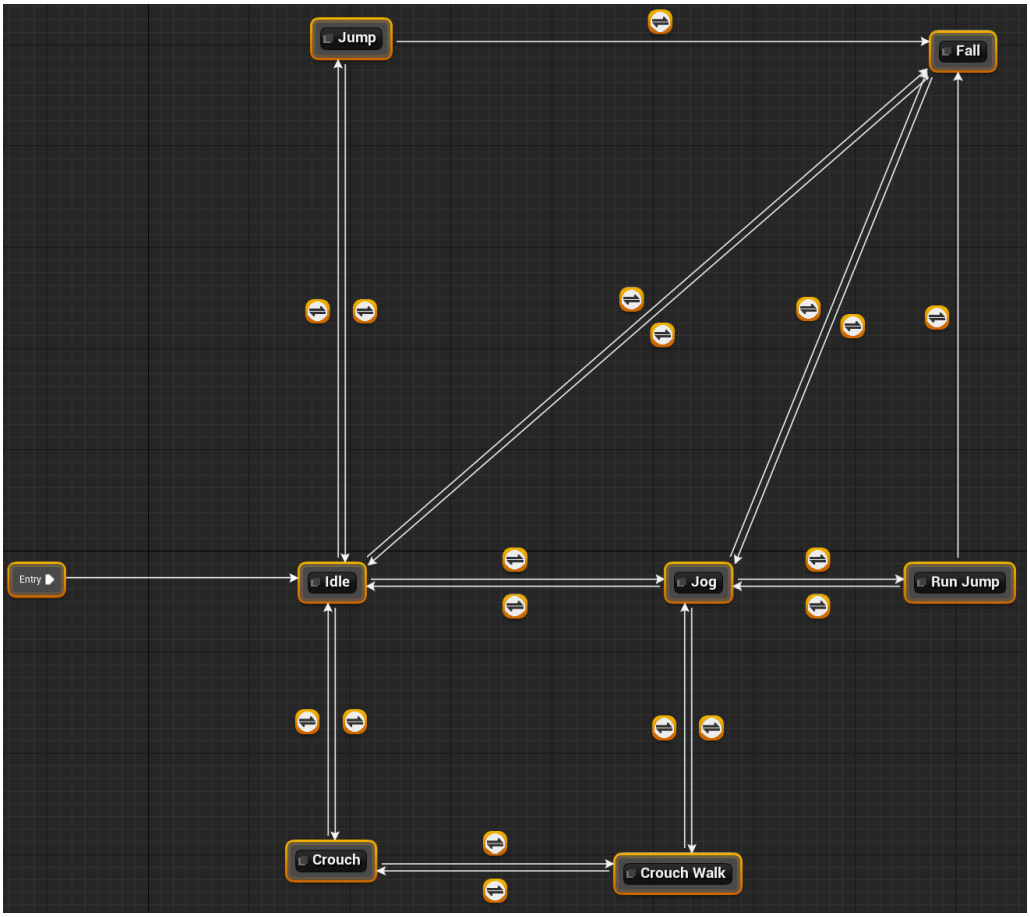
Mostly, the messages have no effect. But with a player character, the message is displayed on the Player's HUD.

Character Animation

Animation for the default third-person character mesh is controlled by the ABP_Character blueprint.

On each animation update, the BP_Character actor is queried for rotation, velocity, and whether jump button and crouch button are down. The Direction, Speed, Horizontal Speed, Falling, Enable Jump, and Crouching variables are set as a result. These variables then control the animation through the state machine ("Locomotion"). After a brief delay, Enable Jump is reset.

STATE MACHINE



IDLE

The entry state. Idle animation.

If Jump is enabled and Horizontal Speed is less than 10, transition to the Jump state.

If Jump is not enabled and Falling is true, transition to the Fall state.

If Horizontal Speed is greater than or equal to 10, transition to the Jog state.

If Crouching is true, transition to the Crouch state.

JOG

Blend Space BS_Jog animation, parametrized by Direction and Horizontal Speed.

If Horizontal Speed is less than 10, transition to the Idle state.

If Horizontal Speed is greater than or equal to 10 and Jump is enabled, transition to the Run Jump state.

If Falling is true and Jump is not enabled, transition to the Fall state.

If Crouching is true, transition to the Crouch Walk state.

RUN JUMP

Jump from Jog animation.

If animation done and falling, transition to Fall state.

If animation done and not falling, transition to Jog state.

FALL

Idle animation.

If not Falling and Horizontal Speed is greater than or equal to 10, transition to Jog state.

If not Falling and Horizontal Speed is less than 10, transition to Idle state.

JUMP

Jump from Stand animation.

If animation finished and falling, transition to Fall state.

If animation finished and not falling, transition to Idle state.

CROUCH

Crouch Idle animation.

If not Crouching, transition to Idle state.

If Horizontal Speed is greater than or equal to 10, transition to Crouch Walk state.

CROUCH WALK

Blend Space BS_CrouchWalk parametrized by Direction and Horizontal Speed.

If not Crouching, transition to Jog state.

If Horizontal Speed is less than 10, transition to Crouch state.

AI

BOT_BEST

Following are definitions of Bot "substates". Numbers mentioned are usually configurable. Blue

WANTS HEALTH

True if health is less than 75%

NEEDS HEALTH

True if health is less than 50%

WANTS AMMO

True if less than 100 hit points worth of ammo left

FOCUSED AMMO IS FULL

True if the Focused Ammo is set and the bot has that weapon and it can take no more ammo

FOCUSED POWERUP IS FULL

True if the Focused Powerup is set and the bot has that powerup and it can take no more charge

NEEDS AMMO

True if less than 50 hit points worth of ammo left

IN MORTAL DANGER

True if health is less than 25%

HAS AMMO

True if has any ammo in any possessed weapon

KNOWS ENEMY LOCATION

True if enemy (i.e. the player) detected by sight, hearing, or being shot at

SUSPECTS ENEMY LOCATION

True if enemy detected less than 5 seconds ago

IN ATTACK RANGE OF ENEMY

True if enemy within range of its current weapon

FOCUSED BOT

Teammate bot that is referred to in the next three boolean variables.

FOCUSED BOT FORWARD

A point 400 Unreal Units forward from the Focused Bot

FOCUSED BOT BACKWARD

A point 400 Unreal Units backward from the Focused Bot

NEAR ANOTHER BOT NEITHER ATTACKING OR FLEEING

If neither Attacking nor Fleeing, nor Needs Ammo nor Needs Health, and suspects location of other bot (on same team), within 4000 Unreal Units of the bot, and the bot is neither attacking or fleeing

NEAR ANOTHER BOT THAT IS ATTACKING

If neither Attacking nor Fleeing, nor Needs Ammo nor Needs Health, and suspects location of other bot (on same team), within a range of 4000 Unreal Units of the bot, and the bot is Attacking

NEAR ANOTHER BOT THAT IS FLEEING

If neither Attacking nor Fleeing, nor Needs Ammo nor Needs Health, and suspects location of other bot (on same team), within 4000 Unreal Units of the bot, and the bot is Fleeing

FOCUSED HEALTH

Health that is referred to in the next two boolean variables

NEAR HEALTH

If Knows Location of Health and health is within 4000 Unreal Units of self

KNOWS LOCATION OF HEALTH

If knows location of health (bot never forgets if has seen the health, or if it is granted sufficient "knowledge of map" even if it hasn't seen the health; but will forget once the health is taken up unless it is a respawnable health)

BORED

In the same state for more than 60 seconds (with some randomness)

FOCUSED WAYPOINT

Waypoint that is referred to in the next boolean variable

NEAR WAYPOINT

If a waypoint is within 4000 Unreal Units of self

FAR FROM HOME

If no waypoint is within 10000 Unreal Units of self and there are waypoints for this bot

FOCUSED AMMO OR WEAPON

Ammo or weapon that is referred to in the next two boolean variables

NEAR AMMO OR WEAPON

If Knows Location of Ammo or Weapon and ammo or weapon is within 4000 Unreal Units of self

KNOWS LOCATION OF AMMO OR WEAPON

If knows location of ammo or weapon (bot never forgets if has seen the pickup, or if it is granted sufficient "knowledge of map" even if it hasn't seen the ammo or weapon; but will forget once the ammo or weapon is taken up unless it is a respawnable ammo or weapon)

FOCUSED POWERUP

Powerup that is referred to in the next two boolean variables

NEAR POWERUP

If Knows Location of Powerup and powerup is within 4000 Unreal Units of self

KNOWS LOCATION OF POWERUP

If knows location of powerup (bot never forgets if has seen the powerup, or if it is granted sufficient "knowledge of map" even if it hasn't seen the powerup; but will forget once the powerup is taken up unless it is a respawnable powerup)

STATES

Here are the states the Bot can be in. The conditionals for transitioning to other states are listed in priority order, that is the first is checked, then if no transition occurs, the second is checked, and so on.

ATTACKING

If Suspects Enemy Location AND (In Mortal Danger OR NOT Has Ammo), transition to Fleeing

If Needs Health and Near Health, transition to Collecting Health

If Needs Ammo and Near Ammo or Weapon and can use that ammo or weapon, transition to Collecting Ammo or Weapon

If NOT Suspects Enemy Location, transition to Wandering

(Action) Move toward enemy unless closer than 400 Unreal units, back up instead; Also move left or right randomly

FLEEING

(Action) Move away from enemy

If Has Ammo AND In Attack Range of Enemy AND Knows Enemy Location AND NOT In Mortal Danger, transition to Attacking

If Needs Health and Near Health, transition to Collecting Health

If Needs Ammo and Near Ammo or Weapon and can use that ammo or weapon, transition to Collecting Ammo or Weapon

After 10 seconds, transition to patrolling

If NOT Suspects Enemy Location, transition to Wandering

PATROLLING

If there are no waypoints, transition to Wandering

(Action) If there is a Focused Waypoint, make that the next waypoint then clear the Focused Waypoint

(Action) go to next waypoint, then increment waypoint

If Needs Health, transition to Finding Health

If Needs Ammo, transition to Finding Ammo or Weapon

If suspects enemy location, transition to Attacking.

If Near Another Bot that is Attacking, transition to Attacking With Teammate

If Near Another Bot that is Fleeing, transition to Defending Teammate

If Wants Health and Near Health, transition to Collecting Health

If Wants Ammo and Near Ammo or Weapon and (Weapon is not Full OR doesn't have weapon), transition to Collecting Ammo or Weapon

If Near Powerup and NOT Focused Powerup is Full, transition to Collecting Powerup

If Bored, transition to Wandering

WANDERING

(Action) Move randomly, avoiding obstacles

If Needs Health, transition to Finding Health

If Needs Ammo, transition to Finding Ammo or Weapon

If suspects enemy location, transition to Attacking.

If Near Another Bot Neither Attacking nor Fleeing, transition to Patrolling

If Near Another Bot that is Attacking, transition to Attacking With Teammate

If Near Another Bot that is Fleeing, transition to Defending Teammate

If Wants Health and Near Health, transition to Collecting Health

If Wants Ammo and Near Ammo or Weapon and can use that ammo or weapon, transition to Collecting Ammo or Weapon

If Near Powerup and can use that powerup, transition to Collecting Powerup

If Far From Home, transition to Patrolling

FINDING HEALTH

If Knows Location of Health, transition to Collecting Health

Otherwise, transition to Wander

FINDING AMMO OR WEAPON

If Knows Location of Ammo or Weapon, transition to Collecting Ammo or Weapon

Otherwise transition to Wander

COLLECTING HEALTH

(Action) Move toward Focused Health

On collection of Health, or failure, transition to Wander

COLLECTING AMMO OR WEAPON

(Action) Move toward Focused Ammo or Weapon

On collection of Ammo or Weapon, or failure, transition to Wander

COLLECTING POWERUP

(Action) Move toward Focused Powerup

On collection of Powerup, or failure, transition to Wander

DEFENDING TEAMMATE

If Suspects Enemy Location AND (In Mortal Danger OR NOT Has Ammo), transition to Fleeing

(Action) Move toward Focused Bot Backward

If Knows Enemy Location, transition to Attacking

If NOT Near Another Bot that is Fleeing, transition to Wandering

ATTACKING WITH TEAMMATE

If Suspects Enemy Location AND (In Mortal Danger OR NOT Has Ammo), transition to Fleeing

(Action) Move toward Focused Bot Forward

If Knows Enemy Location, transition to Attacking

If NOT Near Another Bot that is Attacking, transition to Wandering

INVENTORY

The pickup procedure is roughly as follows: if a pickup is overlapped, or if the player "operates" the pickup, its unique name (or just the regular name in the case of keys) is added to the list of pickups found. Then it is determined whether or not the item can in fact be picked up.

If the player does not already have the item, it is picked up (depending on the pickup, it might be destroyed and its data added to the player rather than the object literally being attached to the player). If the player already has the item but it is of a type that can be depleted (such as ammo or a weapon), the item the player has is replenished with the pickup's amount (if it is positive) and the pickup is destroyed. Otherwise, the pickup is ignored (though if it is a physics body, it may be nudged by the player).

KEY BASE

Health Base

Two properties: Amount (of health to be added when picked up, float), and Super (if true, ignores the maximum of 100 health points, Boolean).

Powerup Base

Ammo Base

Weapon Base

file:///D:/Documents/Repos/Deplorable_Mountaineer/Spreadsheets/Weapon.ods

Also current ammo, last-fire-time

Weapon types include:

Hitscan (beam, fast projectile, or invisible), Projectile (particle or mesh), Bouncing Projectile (grenade), Controllable Projectile, Homing Projectile, Sticky Projectile (Mine), Wide-angle hit (flame thrower).

Effects on target include damage to health and physics. Can add other effects, like stealing powerups.

Comparison of Weapons

file:///D:/Documents/Repos/Deplorable_Mountaineer/Spreadsheets/Weapon_Comparison.ods

PHYSICS MESH BASE

OTHER BLUEPRINT OBJECTS

Characters, Weapons, ammo, pickups, projectiles, Pickup Spawner

Straight Line Fence Builder

Just enter the length (in Unreal Units) of the fence, and the height. Some scaling (as little as possible) will automatically be done to make it exact.

The fence is built from an optional start mesh, then an odd mesh and then an even mesh, then an odd, then an even, and so on to reach the desired length. It can be forced to end with an odd mesh or an even mesh. Finally there is an optional end mesh.

The user can configure the lengths and heights of the various meshes in order to compute how many are needed, and how much scaling is needed, to achieve the configured length and height. These do not have to correspond to actual mesh dimensions (e.g. if there are gaps in the fence).

Stair builder

Enter the number of steps. Default step has a rise and run both of 15. If using a different step mesh, it must have a socket named "Next" for the next step.

To ensure bots can walk up and down the stairs, click the RecastNavMesh actor and change Cell Height to 22.5.

Pipe builder

Fence builder

Hose Builder

Railing builder

Jump pad

Use any actor for jump target (suggested: Target Point actor). Set this in the Target property of the jump pad. If jump pad fails, it could be that there is no solution in the velocity range given, so try changing the min speed and max speed. Use the curved beam to help with positioning (it can be left on or turned off in game--it only gives the approximate path, since it is a spline and not a parabola). The landing area should be at least as large as the jump pad's trigger volume (default radius of 80 so landing area needs diameter of 160 or more) because of inaccuracies due to approaching the pad from different directions.

PLACEABLE SHOOTERS

DAY AND NIGHT SKIES

LIGHTED SIGNS

MONITORS

SECURITY CAMERAS

TELEPORTER

SPAWNER

FAN

SOUNDS

file:///D:/Documents/Repos/Deplorable_Mountaineer/Spreadsheets/Sounds.ods

Guns firing, reloading

Footsteps, thud from fall

Object dropped

Projectile impact

Door open close lock

computery

fans

Elevator

Pickup, injury, death sounds

Powerup use sounds

Menu UI sounds

Fire, Explosion, Steam hiss

Rocket travel

Grenade clink

Running water

Wind

Outdoor ambience

City Ambience

Starter music

Water drip

DEVELOPMENT UTILITIES

BP_Material_Preview

For previewing a material; see the Content/ClassicFPS/Maps/TestMaps/Overview_Materials map for examples.

BP_Physmat_Preview

For previewing a material's physmat properties; see the Content/ClassicFPS/Maps/TestMaps/Overview_Materials map for examples.

BP_Ruler

For measuring distances in a level. Place the BP_Ruler at one place, and place any actor (such as the Target Point actor) at another, then with the BP_Ruler selected, change the Measure From Actor property to the actor you just placed. In the level, the BP_Ruler will display the Z distance (height) from the other actor.

M_ColorGrid

For checking the UV of a mesh. Just apply the material to the mesh you want to check.

M_ColorGrid_TileTest

Another UV-testing material, with irregular tiles so correct tiling can be verified.

CHAPTER 4: PHYSICS

COLLISION

Note, most values are the UE4 Defaults.

file:///D:/Documents/Repos/Deplorable_Mountaineer/Spreadsheets/Collision.ods

PHYSICAL SURFACES

These values are for game purposes--in some cases I just made them up.

file:///D:/Documents/Repos/Deplorable_Mountaineer/Spreadsheets/PhysicalSurfaces.ods

CHAPTER 5: UTILITY

CHAPTER 6: TUTORIALS

If you get stuck in any of these tutorials, the complete solutions can be found in the Content Browser, in the folders Content->ClassicFPS->Tutorial_Solutions->Tutorials_[*num*].

TUTORIAL 1: SET UP PROJECT FOR TUTORIALS (BEGINNER)

Before beginning this tutorial, you should understand the basics of Unreal Engine 4. To ensure this, it is recommended that inexperienced users complete the *Level Designer Quick Start* tutorial provided by Epic and found at https://docs.unrealengine.com/latest/INT/Engine/QuickStart/index.html.

1. Create and open a new *Classic FPS* project.
   A. Open the *Epic Games Launcher*.
   B. Navigate to the *Vault* section.
      i. Click to select the *Unreal Engine* tab at the top.
      ii. Click to select the *Library* tab along the left side.
      iii. Scroll down and find the *Vault* section.
   C. In the *Vault*, find the Classic FPS pack and click *Create Project*.
   D. In the popup bar, either leave the defaults or change the name, directory, and unreal version to your liking. You might wish to change the name to something like ClassicFPS_Tutorial.
   E. Click *Create*.
   F. Give the software time to create the project.
   G. When done, scroll back up to the *My Projects* section and find your newly-created project.
   H. Double click on the project to open it and wait for it to open.
2. In the *Content Browser* panel (by default, on the lower left of the main window) navigate to the top level *Content* section by clicking on *Content* in the breadcrumbs across the top of the *Content Browser* panel. You will see a *ClassicFPS* folder there, where all the Classic FPS material is kept.
3. Create and open a new folder called *Tutorials*.
   A. Right-click a blank area of the *Content Browser*.
   B. Select *New Folder* in the popup that appears. A new folder called *NewFolder* will be created, with the name highlighted for editing.
   C. Change the name of the new folder to *Tutorials*.
      i. If for some reason, the folder name is no longer highlighted for editing, just select the folder and type *F2*.
   D. Double click the newly-created Tutorials folder to open it.
4. Create and enter a new map called *TutorialMap*.
   A. Right-click a blank area of the *Content Browser*.
   B. Select *Level* in the popup that appears. A new level called *NewWorld* will be created, with the name highlighted for editing.
   C. Change the name of the new level to *TutorialMap*.
      i. If for some reason, the level name is no longer highlighted for editing, just select the level and type *F2*.
   D. Double click the newly-created *TutorialMap* level to open it. A blank level will appear in the viewport.

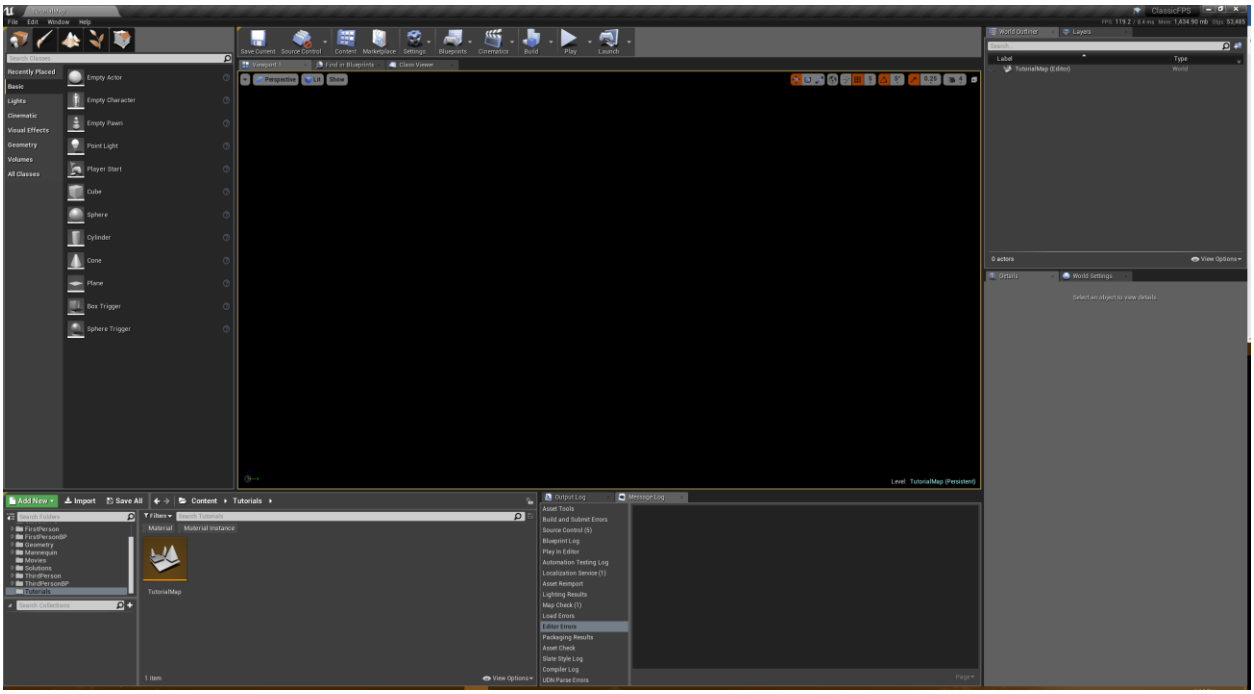


FIGURE 2: NEW BLANK LEVEL CREATED IN THE NEW PROJECT

5. Add a *Directional Light* to the level.
   A. In the *Modes* panel (by default, at the top left of the main window), make sure you are in *Place Mode* (the icon with a cube and a lightbulb are selected). Clicking *Shift+1* will do that.
   B. Select the *Lights* tab on the left side to show all light assets.
   C. Click and drag the *Directional Light* asset into the viewport. A *Sun* icon, representing the *Directional Light*, will appear in the viewport.
   D. Making sure the *Directional Light* is still selected, change its position to (0, 0, 1000).
      i. Find the *Details Panel*, by default on the lower-right side of the main window.
      ii. Find the *Transform* section in the *Details Panel*.
      iii. Find the *Location* property in the *Transform* section of the *Details Panel*.
      iv. Change the red *X* field to 0, the green *Y* field to 0, and the blue *Z* field to 1000.
   E. Making sure the *Directional Light* is still selected, change its rotation to (40, -45, 0).
      i. Find the *Details Panel*, by default on the lower-right side of the main window.
      ii. Find the *Transform* section in the *Details Panel*.
      iii. Find the *Rotation* property in the *Transform* section of the *Details Panel*.
      iv. Change the red *X* field to 40, the green *Y* field to -45, and the blue *Z* field to 0.
6. Add a platform representing the ground to the level.
   A. In the *Modes* panel (by default, at the top left of the main window), make sure you are in *Place Mode* (the icon with a cube and a lightbulb are selected). Clicking *Shift+1* will do that.
   B. Select the *Geometry* tab on the left side to show all *Geometry* assets.
   C. Click and drag the *Box* asset into the viewport. A cube will appear in the viewport.
   D. Making sure the *Box* is still selected, change its position to (0, 0, 0).
      i. Find the *Details Panel*, by default on the lower-right side of the main window.

ii. Find the *Transform* section in the *Details Panel*.
iii. Find the *Location* property in the *Transform* section of the *Details Panel*.
iv. Change the red *X* field to 0, the green *Y* field to 0, and the blue *Z* field to 0.

E. Change the box's dimensions to 5000x5000x100.
i. Find the *Details Panel*, by default on the lower-right side of the main window.
ii. Find the *Brush Settings* section in the *Details Panel*.
iii. Find the *X*, *Y*, and *Z* properties in the *Brush Settings* section of the *Details Panel*.
iv. Change the *X* field to 5000, the *Y* field to 5000, and the *Z* field to 100.

F. With the box (the ground) selected, give it a *Grass* material.
i. Click on the surface of the ground to ensure the top surface of the box is selected.
ii. Find the *Details Panel*, by default on the lower-right side of the main window.
iii. Find the *Surface Materials* section in the *Details Panel*.
iv. Find the *Element 0* property in the *Surface Materials* section of the *Details Panel*.
v. Click the box that says *None* with a downward-pointing triangular arrow to the right.
vi. Start typing *MI_Grass_Patchy* in the *Search Assets* bar of the dropdown that appears.
vii. When you see the *MI_Grass_Patchy* material appear in the dropdown, select it. The ground will now be covered with a dirt and grass patchy material.

7. Add a sky.

A. In the *Modes* panel (by default, at the top left of the main window), make sure you are in *Place Mode* (the icon with a cube and a lightbulb are selected). Clicking *Shift+1* will do that.
B. Click the *Search Classes* search bar near the top of the *Modes* panel.
C. In the search bar, start typing *BP_Sky_Sphere*.
D. When you see the *BP_Sky_Sphere* asset, Click and drag it into the viewport. A red sky will now appear.
E. Making sure the sky is still selected, change its position to (0, 0, 0).
i. Find the *Details Panel*, by default on the lower-right side of the main window.
ii. Find the *Transform* section in the *Details Panel*.
iii. Find the *Location* property in the *Transform* section of the *Details Panel*.
iv. Change the red *X* field to 0, the green *Y* field to 0, and the blue *Z* field to 0.
F. Wire the sky to the sun.
i. Make sure the sky is still selected.
ii. Find the *Details Panel*, by default on the lower-right side of the main window.
iii. Find the *Default* section in the *Details Panel*.
iv. Find the *Directional Light Actor* property in the *Default* section in the *Details Panel*.
v. Click the eyedropper icon to the right.
vi. In the viewport, find the *Directional Light Actor* you had placed before, and click on it. The sky should turn blue and the sun will now be in the right place in the sky to shine light in the direction of the *Directional Light*.

8. Add a skylight actor.

A. In the *Modes* panel (by default, at the top left of the main window), make sure you are in *Place Mode* (the icon with a cube and a lightbulb are selected). Clicking *Shift+1* will do that.
B. If the search bar is still active, click the *X* in it to clear it.
C. Select the *Lights* tab on the left side to show all light assets.

D. Click and drag the *Sky Light* asset into the viewport. A *Hemisphere Light* icon, representing the *Sky Light*, will appear in the viewport.
E. Making sure the *Sky Light* is still selected, change its position to (0, 0, 1000).
    i. Find the *Details Panel*, by default on the lower-right side of the main window.
    ii. Find the *Transform* section in the *Details Panel*.
    iii. Find the *Location* property in the *Transform* section of the *Details Panel*.
    iv. Change the red *X* field to 0, the green *Y* field to 0, and the blue *Z* field to 1000.
F. Making sure the *Sky Light* is still selected, change its mobility to *Static*.
    i. Find the *Details Panel*, by default on the lower-right side of the main window.
    ii. Find the *Transform* section in the *Details Panel*.
    iii. Find the *Mobility* property in the *Transform* section of the *Details Panel*.
    iv. Change the property to *Static*.

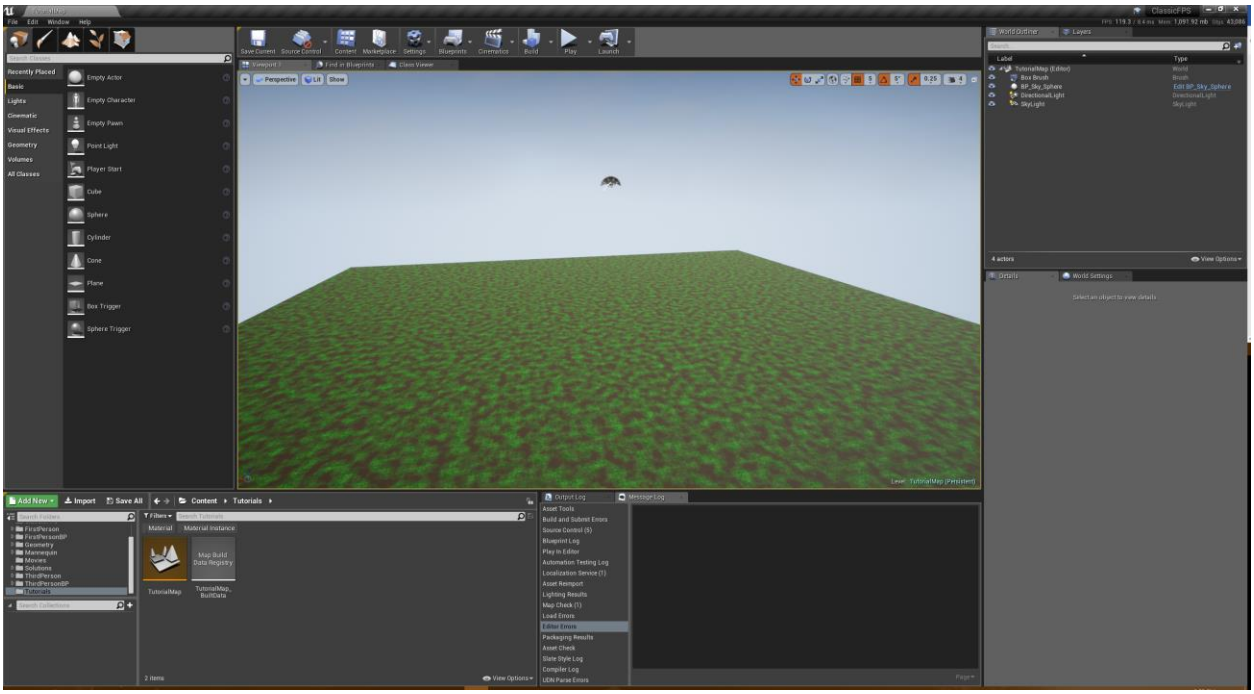
9. Build the lighting.

A. In the toolbar, by default at the top center of the main window, click on the *Build* icon (looks like a cluster of buildings). Wait for it to finish. There should be no errors or warnings.

10. Save the level.

A. In the toolbar, by default at the top center of the main window, click on the *Save* icon (looks like a 3.5in floppy disk). A new auxiliary asset called *TutorialMap_BuiltData* will appear in the *Content Browser* panel.

You now have a basic map to do the other tutorials in.



TUTORIAL 2: ADD A SWITCHABLE LIGHT (EASY)

1. Open the *Tutorial Map* created in *Tutorial 1*.
   A. In the *Content Browser*, navigate to Content->Tutorials and double click *TutorialMap*.
   B. If the file is not there (for example, if you skipped *Tutorial 2* or a different tutorial was recently completed) copy it from Content->ClassicFPS->Tutorial_Solutions->Tutorial_1 to Content->Tutorials.
      i. If necessary, create the new folder *Tutorials* in the top-level *Content* folder in the *Content Browser*.
      ii. Ensure the *Sources* panel of the *Content Browser* is showing.
         a. If you do not see a directory hierarchy to the left of the folders view of the *Content Browser* panel, click the icon just under the *Add New* button that has a white right-pointing triangular arrow on it. It will open the *Sources* panel which shows directory names
      iii. Scroll to make sure the *Tutorials* directory is visible.
      iv. In the folder panel on the right side of the *Content Browser*, navigate (if necessary) to Content->ClassicFPS->Tutorial_Solutions->Tutorial_1.
      v. Click and drag the *TutorialMap* asset to the left and carefully drop it onto the *Tutorials* directory. A popup menu will appear.
      vi. In the popup, select *Copy Here*.
      vii. Navigate to Content->Tutorials and make sure you successfully copied *TutorialMap* there.
      viii. Now you can double-click this *TutorialMap* icon and open a copy of the level.
2. Create a small room for the light and switch.
   A. In the *Modes* panel (by default, at the top left of the main window), make sure you are in *Place Mode* (the icon with a cube and a lightbulb are selected). Clicking *Shift+1* will do that.
   B. Select the *Geometry* tab on the left side to show all *Geometry* assets.
   C. Click and drag the *Box* asset into the viewport. A cube will appear in the viewport.
   D. Making sure the *Box* is still selected, change its position to (0, 0, 200).
      i. Find the *Details Panel*, by default on the lower-right side of the main window.
      ii. Find the *Transform* section in the *Details Panel*.
      iii. Find the *Location* property in the *Transform* section of the *Details Panel*.
      iv. Change the red *X* field to 0, the green *Y* field to 0, and the blue *Z* field to 200.
   E. Change the box's dimensions to 300x300x300.
      i. Find the *Details Panel*, by default on the lower-right side of the main window.
      ii. Find the *Brush Settings* section in the *Details Panel*.
      iii. Find the *X*, *Y*, and *Z* properties in the *Brush Settings* section of the *Details Panel*.
      iv. Change the *X* field to 300, the *Y* field to 300, and the *Z* field to 300.
   F. Make the box hollow.
      i. Find the *Details Panel*, by default on the lower-right side of the main window.
      ii. Find the *Brush Settings* section in the *Details Panel*.
      iii. Find the *Hollow* property in the *Brush Settings* section of the *Details Panel*.
      iv. Check the box to activate the *Hollow* property.
   G. Cut a doorway into the box.
      i. In the *Modes* panel (by default, at the top left of the main window), make sure you are in *Place Mode* (the icon with a cube and a lightbulb are selected). Clicking *Shift+1* will do that.

ii. Select the *Geometry* tab on the left side to show all *Geometry* assets.
iii. Click and drag another *Box* asset into the viewport. A cube will appear in the viewport.
iv. Making sure the *Box* is still selected, change its position to (-145, 0, 155).
    a. Find the *Details Panel*, by default on the lower-right side of the main window.
    b. Find the *Transform* section in the *Details Panel*.
    c. Find the *Location* property in the *Transform* section of the *Details Panel*.
    d. Change the red *X* field to -145, the green *Y* field to 0, and the blue *Z* field to 155.
v. Change the box's dimensions to 20x100x200.
    a. Find the *Details Panel*, by default on the lower-right side of the main window.
    b. Find the *Brush Settings* section in the *Details Panel*.
    c. Find the *X*, *Y*, and *Z* properties in the *Brush Settings* section of the *Details Panel*.
    d. Change the *X* field to 20, the *Y* field to 100, and the *Z* field to 200.
vi. Make the box into a subtractive brush.
    a. Find the *Details Panel*, by default on the lower-right side of the main window.
    b. Find the *Brush Settings* section in the *Details Panel*.
    c. Find the *Brush Type* property in the *Brush Settings* section of the *Details Panel*.
    d. Click the property and change it from *Additive* to *Subtractive*. The box will become invisible and you will now have a doorway into the room.

H. Build the geometry by clicking the Build icon (looks like a cluster of buildings) in the toolbar above the Viewport. Make sure there are no errors or warnings. The room will now be dark inside.
I. Test by clicking the Play button (a right-pointing triangle icon) in the toolbar above the Viewport. Make sure the character can walk through the door.

3. Place a light switch to the right of the door just outside the room.

A. Navigate to Content->ClassicFPS->Environment->Props->Switches in the *Content Browser*.
B. Drag the *BP_LightSwitch* asset into the *Viewport*.
C. Making sure the *BP_LightSwitch* actor is still selected, change its position to (-150, 65, 180).
    i. Find the *Details Panel*, by default on the lower-right side of the main window.
    ii. Find the *Transform* section in the *Details Panel*.
    iii. Find the *Location* property in the *Transform* section of the *Details Panel*.
    iv. Change the red *X* field to -150, the green *Y* field to 65, and the blue *Z* field to 180.
D. Making sure the *BP_LightSwitch* actor is still selected, change its rotation to (0, 0, 180).
    i. Find the *Details Panel*, by default on the lower-right side of the main window.
    ii. Find the *Transform* section in the *Details Panel*.
    iii. Find the *Rotation* property in the *Transform* section of the *Details Panel*.
    iv. Change the red *X* field to 0, the green *Y* field to 0, and the blue *Z* field to 180.
E. Test the switch.
    i. Play the level by clicking the *Play* icon in the toolbar.
    ii. Using the (by default) W, A, S, and D keys to move and mouse to turn, walk up to the switch and look directly at it (the green crosshair should be right on the yellow toggle on the switch and the character should be as close to the switch as possible).
    iii. Press the Operate key (by default, E or F) and check that the switch can be flipped on and off.

4. Place a ceiling light at the center of the ceiling of the room.

A. Navigate to Content->ClassicFPS->Environment->Props->Lights in the *Content Browser*.

B. Drag the *BP_CeilingLamp* asset into the *Viewport*.
C. Making sure the *BP_CeilingLamp* actor is still selected, change its position to (0, 0, 345).
    i. Find the *Details Panel*, by default on the lower-right side of the main window.
    ii. Find the *Transform* section in the *Details Panel*.
    iii. Find the *Location* property in the *Transform* section of the *Details Panel*.
    iv. Change the red *X* field to 0, the green *Y* field to 0, and the blue *Z* field to 345.

5. Wire the switch to the light.
A. Select the *BP_LightSwitch* actor in the *Viewport*.
B. Find the *Details Panel*, by default on the lower-right side of the main window.
C. Find the *State Machine* section in the *Details Panel*.
D. Find the *Forward Actions to* property in the *State Machine* section of the *Details Panel*.
E. Click the Plus sign to the right of the property to add a new element to the array. Element 0, with value *None*, should appear below the property.
F. Click on the eyedropper to the right of the Element 0 value.
G. Then in the Viewport, navigate to and click on the ceiling lamp. The *None* value should change to *BP_CeilingLamp*.
H. Test the switch and the light.
    i. Play the level by clicking the *Play* icon in the toolbar.
    ii. Using the (by default) W, A, S, and D keys to move and mouse to turn, walk up to the switch and look directly at it (the green crosshair should be right on the yellow toggle on the switch and the character should be as close to the switch as possible).
    iii. Press the Operate key (by default, E or F) and check that as the switch is flipped on and off, the ceiling light turns on and off.

6. Build the lighting.
A. In the toolbar, by default at the top center of the main window, click on the *Build* icon (looks like a cluster of buildings). Wait for it to finish. There should be no errors or warnings.

7. Save the level.
A. In the toolbar, by default at the top center of the main window, click on the *Save* icon (looks like a 3.5in floppy disk). A new auxiliary asset called *TutorialMap_BuiltData* will appear in the *Content Browser* panel.

TUTORIAL 3: ADD A LOCKABLE DOOR (MEDIUM EASY)

1. Open the *Tutorial Map* created in *Tutorial 2*.
   A. In the *Content Browser*, navigate to Content->Tutorials and double click *TutorialMap*.
   B. If the correct file is not there (for example, if you skipped *Tutorial 2* or a different tutorial was recently completed) copy it from Content->ClassicFPS->Tutorial_Solutions->Tutorial_2 to Content->Tutorials.
      i. If necessary, create the new folder *Tutorials* in the top-level *Content* folder in the *Content Browser*.
      ii. Ensure the *Sources* panel of the *Content Browser* is showing.
         a. If you do not see a directory hierarchy to the left of the folders view of the *Content Browser* panel, click the icon just under the *Add New* button that has a white right-pointing triangular arrow on it. It will open the *Sources* panel which shows directory names
      iii. Scroll to make sure the *Tutorials* directory is visible.
      iv. In the folder panel on the right side of the *Content Browser*, navigate (if necessary) to Content->ClassicFPS->Tutorial_Solutions->Tutorial_2.
      v. Click and drag the *TutorialMap* asset to the left and carefully drop it onto the *Tutorials* directory. A popup menu will appear.
      vi. In the popup, select *Copy Here*.
      vii. Navigate to Content->Tutorials and make sure you successfully copied *TutorialMap* there.
      viii. Now you can double-click this *TutorialMap* icon and open a copy of the level.
2. Place a door in the doorway.
   A. Navigate to Content->ClassicFPS->Environment->Buildings->Residential in the *Content Browser*.
   B. Drag the *BP_Door* asset into the *Viewport*.
   C. Making sure the *BP_Door* actor is still selected, change its position to (-150, 0, 55).
      i. Find the *Details Panel*, by default on the lower-right side of the main window.
      ii. Find the *Transform* section in the *Details Panel*.
      iii. Find the *Location* property in the *Transform* section of the *Details Panel*.
      iv. Change the red *X* field to -150, the green *Y* field to 0, and the blue *Z* field to 55.
   D. Test the door.
      i. Play the level by clicking the *Play* icon in the toolbar.
      ii. Using the (by default) W, A, S, and D keys to move and mouse to turn, walk up to the door—it should open inward. If you back off, it should auto-close in 5 seconds. If you approach the door from inside the room, it should open outward. You can also operate the door (with the E or F key by default) to open and close it.
3. Place a lock on the wall to the left of the door handle
   A. Navigate to Content->ClassicFPS->Environment->Props->Switches in the *Content Browser*.
   B. Drag the *BP_DoorLock* asset into the *Viewport*.
   C. Making sure the *BP_DoorLock* actor is still selected, change its position to (-150, -60, 156).
      i. Find the *Details Panel*, by default on the lower-right side of the main window.
      ii. Find the *Transform* section in the *Details Panel*.
      iii. Find the *Location* property in the *Transform* section of the *Details Panel*.
      iv. Change the red *X* field to -150, the green *Y* field to 60, and the blue *Z* field to 145.

D. Making sure the *BP_DoorLock* actor is still selected, change its rotation to (0, 0, 180).
   i. Find the *Details Panel*, by default on the lower-right side of the main window.
   ii. Find the *Transform* section in the *Details Panel*.
   iii. Find the *Rotation* property in the *Transform* section of the *Details Panel*.
   iv. Change the red *X* field to 0, the green *Y* field to 0, and the blue *Z* field to 180.

E. Test the lock.
   i. Play the level by clicking the *Play* icon in the toolbar.
   ii. Using the (by default) W, A, S, and D keys to move and mouse to turn, walk up to the lock and look directly at it (the green crosshair should be right on the touchpad and the character should be as close to the switch as possible).
   iii. Press the Operate key (by default, E or F) and check that as the lock is swapped between red (locked) and green (unlocked).

4. Wire the lock to the door.

A. Select the *BP_DoorLock* actor in the *Viewport*.
B. Find the *Details Panel*, by default on the lower-right side of the main window.
C. Find the *State Machine* section in the *Details Panel*.
D. Find the *Forward Actions to* property in the *State Machine* section of the *Details Panel*.
E. Click the Plus sign to the right of the property to add a new element to the array. Element 0, with value *None*, should appear below the property.
F. Click on the eyedropper to the right of the Element 0 value.
G. Then in the Viewport, navigate to and click on the door. The *None* value should change to *BP_Door*.
H. Now, make sure the door starts out in the same state as the lock, i.e. locked.
   i. Select the *BP_Door* actor in the *Viewport*.
   ii. Find the *Details Panel*, by default on the lower-right side of the main window.
   iii. Find the *State Machine* section in the *Details Panel*.
   iv. Find the *Current State* property in the *State Machine* section of the *Details Panel*.
   v. Change the state from *Closed* to *Closed and Locked* (spelling has to be exact).
I. Test the lock and the door.
   i. Play the level by clicking the *Play* icon in the toolbar.
   ii. You should not be able to open the door at first.
   iii. Using the (by default) W, A, S, and D keys to move and mouse to turn, walk up to the lock and look directly at it (the green crosshair should be right on the touchpad and the character should be as close to the switch as possible).
   iv. Press the Operate key (by default, E or F) and check that as the lock is swapped between red (locked) and green (unlocked).
   v. Confirm you can open the door when unlocked.
   vi. Lock the door again and confirm it is indeed locked.

5. Make the Lock Require a Key to Unlock

A. Select the *BP_DoorLock* actor in the *Viewport*.
B. Find the *Details Panel*, by default on the lower-right side of the main window.
C. Find the *Lock* section in the *Details Panel*.
D. Find the *Key Needed* property in the *Lock* section of the *Details Panel*.
E. Change the field from *None* to *BP_Key_Blue*. The lock will now display a blue key icon on it.

F. Test the Lock. This time it should not unlock but you should receive a text message that a "Blue key" is needed.

6. Add a Blue Key Spawner to the level.

A. Navigate to Content->ClassicFPS->Pickups->Keys in the *Content Browser*.
B. Drag the *BP_Spawner_Key_Blue* asset into the *Viewport*.
C. Making sure the *BP_Spawner_Key_Blue* actor is still selected, change its position to (0, -345, 50).
   i. Find the *Details Panel*, by default on the lower-right side of the main window.
   ii. Find the *Transform* section in the *Details Panel*.
   iii. Find the *Location* property in the *Transform* section of the *Details Panel*.
   iv. Change the red *X* field to 0, the green *Y* field to -345, and the blue *Z* field to 50.
D. Making sure the *BP_Spawner_Key_Blue* actor is still selected, change its rotation to (0, 0, 180).
   i. Find the *Details Panel*, by default on the lower-right side of the main window.
   ii. Find the *Transform* section in the *Details Panel*.
   iii. Find the *Rotation* property in the *Transform* section of the *Details Panel*.
   iv. Change the red *X* field to 0, the green *Y* field to 0, and the blue *Z* field to 180.
E. Test. The lock cannot be operated without the key, but if you pick up the blue key, you can now operate the lock.

7. Build the lighting.

A. In the toolbar, by default at the top center of the main window, click on the *Build* icon (looks like a cluster of buildings). Wait for it to finish. There should be no errors or warnings.

8. Save the level.

A. In the toolbar, by default at the top center of the main window, click on the *Save* icon (looks like a 3.5in floppy disk). A new auxiliary asset called *TutorialMap_BuiltData* will appear in the *Content Browser* panel.

TUTORIAL 4: UNDERWATER AREA IN LEVEL (INTERMEDIATE)

1. Open the *Tutorial Map* created in *Tutorial 3*.
   A. In the *Content Browser*, navigate to Content->Tutorials and double click *TutorialMap*.
   B. If the correct file is not there (for example, if you skipped *Tutorial 3* or a different tutorial was recently completed) copy it from Content->ClassicFPS->Tutorial_Solutions->Tutorial_3 to Content->Tutorials.
      i. If necessary, create the new folder *Tutorials* in the top-level *Content* folder in the *Content Browser*.
      ii. Ensure the *Sources* panel of the *Content Browser* is showing.
         a. If you do not see a directory hierarchy to the left of the folders view of the *Content Browser* panel, click the icon just under the *Add New* button that has a white right-pointing triangular arrow on it. It will open the *Sources* panel which shows directory names
      iii. Scroll to make sure the *Tutorials* directory is visible.
      iv. In the folder panel on the right side of the *Content Browser*, navigate (if necessary) to Content->ClassicFPS->Tutorial_Solutions->Tutorial_3.
      v. Click and drag the *TutorialMap* asset to the left and carefully drop it onto the *Tutorials* directory. A popup menu will appear.
      vi. In the popup, select *Copy Here*.
      vii. Navigate to Content->Tutorials and make sure you successfully copied *TutorialMap* there.
      viii. Now you can double-click this *TutorialMap* icon and open a copy of the level.
2. Create a swimming pool.
   A. In the *Modes* panel (by default, at the top left of the main window), make sure you are in *Place Mode* (the icon with a cube and a lightbulb are selected). Clicking *Shift+1* will do that.
   B. Select the *Geometry* tab on the left side to show all *Geometry* assets.
   C. Click and drag the *Box* asset into the viewport. A cube will appear in the viewport.
   D. Making sure the *Box* is still selected, change its position to (0, 900, -145).
      i. Find the *Details Panel*, by default on the lower-right side of the main window.
      ii. Find the *Transform* section in the *Details Panel*.
      iii. Find the *Location* property in the *Transform* section of the *Details Panel*.
      iv. Change the red *X* field to 0, the green *Y* field to 900, and the blue *Z* field to -145.
   E. Change the box's dimensions to 500x1000x400.
      i. Find the *Details Panel*, by default on the lower-right side of the main window.
      ii. Find the *Brush Settings* section in the *Details Panel*.
      iii. Find the *X*, *Y*, and *Z* properties in the *Brush Settings* section of the *Details Panel*.
      iv. Change the *X* field to 500, the *Y* field to 1000, and the *Z* field to 400.
   F. Carve the pool out of the box.
      i. In the *Modes* panel (by default, at the top left of the main window), make sure you are in *Place Mode* (the icon with a cube and a lightbulb are selected). Clicking *Shift+1* will do that.
      ii. Select the *Geometry* tab on the left side to show all *Geometry* assets.
      iii. Click and drag another *Box* asset into the viewport. A cube will appear in the viewport.
      iv. Making sure the *Box* is still selected, change its position to (0, 900, -140).
         a. Find the *Details Panel*, by default on the lower-right side of the main window.
         b. Find the *Transform* section in the *Details Panel*.

c. Find the *Location* property in the *Transform* section of the *Details Panel*.
d. Change the red *X* field to 0, the green *Y* field to 900, and the blue *Z* field to -140.

v. Change the box's dimensions to 490x990x400.
a. Find the *Details Panel*, by default on the lower-right side of the main window.
b. Find the *Brush Settings* section in the *Details Panel*.
c. Find the *X*, *Y*, and *Z* properties in the *Brush Settings* section of the *Details Panel*.
d. Change the *X* field to 490, the *Y* field to 990, and the *Z* field to 400.

vi. Make the box into a subtractive brush.
a. Find the *Details Panel*, by default on the lower-right side of the main window.
b. Find the *Brush Settings* section in the *Details Panel*.
c. Find the *Brush Type* property in the *Brush Settings* section of the *Details Panel*.
d. Click the property and change it from *Additive* to *Subtractive*. The box will become invisible and you will now have a cavity in the pool.

3. Fill the pool with water.

A. Add a physics volume.
i. In the *Modes* panel (by default, at the top left of the main window), make sure you are in *Place Mode* (the icon with a cube and a lightbulb are selected). Clicking *Shift+1* will do that.
ii. Select the *Volumes* tab on the left side to show all *Volumes* assets.
iii. Click and drag the *Physics Volume* asset into the viewport. A cube will appear in the viewport.
iv. Making sure the *Box* is still selected, change its position to (0, 900, -145).
a. Find the *Details Panel*, by default on the lower-right side of the main window.
b. Find the *Transform* section in the *Details Panel*.
c. Find the *Location* property in the *Transform* section of the *Details Panel*.
d. Change the red *X* field to 0, the green *Y* field to 900, and the blue *Z* field to -145.
v. Change the box's dimensions to 490x990x390.
a. Find the *Details Panel*, by default on the lower-right side of the main window.
b. Find the *Brush Settings* section in the *Details Panel*.
c. Find the *X*, *Y*, and *Z* properties in the *Brush Settings* section of the *Details Panel*.
d. Change the *X* field to 490, the *Y* field to 990, and the *Z* field to 390.
vi. Making sure the *Physics Volume* actor is still selected, make it into a water volume.
a. Find the *Details Panel*, by default on the lower-right side of the main window.
b. Find the *Character Movement* section in the *Details Panel*.
c. Find the *Water Volume* property in the *Character Movement* section of the *Details Panel*.
d. Make sure the box is checked.

B. Add a Post Process Volume
i. In the *Modes* panel (by default, at the top left of the main window), make sure you are in *Place Mode* (the icon with a cube and a lightbulb are selected). Clicking *Shift+1* will do that.
ii. Select the *Volumes* tab on the left side to show all *Volumes* assets.
iii. Click and drag the *Post Process Volume* asset into the viewport. A cube will appear in the viewport.
iv. Making sure the *Box* is still selected, change its position to (0, 900, -145).
a. Find the *Details Panel*, by default on the lower-right side of the main window.

b. Find the *Transform* section in the *Details Panel*.
c. Find the *Location* property in the *Transform* section of the *Details Panel*.
d. Change the red *X* field to 0, the green *Y* field to 900, and the blue *Z* field to -145.

v. Change the box's dimensions to 490x990x390.
a. Find the *Details Panel*, by default on the lower-right side of the main window.
b. Find the *Brush Settings* section in the *Details Panel*.
c. Find the *X*, *Y*, and *Z* properties in the *Brush Settings* section of the *Details Panel*.
d. Change the *X* field to 490, the *Y* field to 990, and the *Z* field to 390.

vi. Making sure the *Post Process Volume* actor is still selected, adjust the Lens settings.
a. Find the *Details Panel*, by default on the lower-right side of the main window.
b. Find the *Lens* section in the *Details Panel*.
c. Find the *Image Effects* drawer in the *Lens* section of the *Details Panel*. If necessary, click the right-pointing white triangular arrow to open it.
d. Check the boxes to enable the fields: *Chromatic Aberration*, *Vignette Intensity*, and *Grain Intensity*.
e. Change *Chromatic Aberration* to 1.5*, Vignette Intensity* to .6, and *Grain Intensity* to .25.
f. Find the *Depth of Field1* drawer in the *Lens* section of the *Details Panel*. If necessary, click the right-pointing white triangular arrow to open it.
g. Check the boxes to enable the fields: *Method*, *Focal Distance*, *Focal Region*, *Near Transition Region*, and *Far Transition Region*.
h. Change *Method* to *GaussianDOF*, *Focal Distance* to 200, *Focal Region* to 200, Near Transition Region to 500, *Far Transition Region* to 1000.

vii. Making sure the *Post Process Volume* actor is still selected, adjust the *Color Grading* settings.
a. Find the *Details Panel*, by default on the lower-right side of the main window.
b. Find the *Color Grading* section in the *Details Panel*.
c. Find the *Global* drawer in the *Color Grading* section of the *Details Panel*.
d. Click the white triangular arrow to open the *Global* drawer.
e. Check the box to enable *Scene Color Tint*.
f. Click the white bar to open the color chooser.
g. Change *R* (red) to .2, *G* (green) to .3, and *B* (blue) to .3.
h. Click *OK*. If you move into the volume, it will appear blue, blurry, and grainy.

viii. Making sure the *Post Process Volume* actor is still selected, adjust the *Post Process Volume Settings*.
a. Find the *Details Panel*, by default on the lower-right side of the main window.
b. Find the *Post Process Volume Settings* section in the *Details Panel*.
c. Find the *Blend Radius* property in the *Post Process Volume Settings* section of the *Details Panel*.
d. Change the *Blend Radius* to 5.

C. Add a water surface.
i. In the *Modes* panel (by default, at the top left of the main window), make sure you are in *Place Mode* (the icon with a cube and a lightbulb are selected). Clicking *Shift+1* will do that.
ii. Select the *Basic* tab on the left side to show all *Basic* assets.
iii. Click and drag the *Plane* asset into the viewport. A plane will appear in the viewport.

iv. Making sure the *Plane* is still selected, change its position to (0, 900, 50).
    a. Find the *Details Panel*, by default on the lower-right side of the main window.
    b. Find the *Transform* section in the *Details Panel*.
    c. Find the *Location* property in the *Transform* section of the *Details Panel*.
    d. Change the red *X* field to 0, the green *Y* field to 900, and the blue *Z* field to 50.
v. Change the plane's scale to 5x10x1.
    a. Find the *Details Panel*, by default on the lower-right side of the main window.
    b. Find the *Transform* section in the *Details Panel*.
    c. Find the *Scale* property in the *Transform* section of the *Details Panel*.
    d. Change the *X* field to 5, the *Y* field to 10, and the *Z* field to 1. (important: the lock to the right must be unlocked. If not, click on it to unlock before setting the scale).
vi. Change the plane's material to Water.
    a. Find the *Details Panel*, by default on the lower-right side of the main window.
    b. Find the *Materials* section in the *Details Panel*.
    c. Find the *Element 0* property in the *Materials* section of the *Details Panel*.
    d. Click the box that says *BasicShapeMaterial* and start typing *M_Water*. When you see the *M_Water* material appear, click it.
vii. Change the plane's collision to *No Collision*.
    a. Find the *Details Panel*, by default on the lower-right side of the main window.
    b. Find the *Collision* section in the *Details Panel*.
    c. Find the *Collision Presets* property in the *Collision* section of the *Details Panel*.
    d. Change the preset from *Default* to *NoCollision*.

D. Test the level. Try jumping in the pool, swimming under water (use the Swim Up and Swim Down keys, by default Space Bar and C key, respectively), and then getting back out onto land.

TUTORIAL 5: ELEVATOR (MEDIUM EASY)

TUTORIAL 6: SECURITY CAMERA(MEDIUM EASY)

TUTORIAL 7: SECRET AREA (MEDIUM EASY)

TUTORIAL 8: END OF LEVEL (MEDIUM EASY)

TUTORIAL 9: CONVEYOR BELT (MEDIUM EASY)

TUTORIAL 10: CRUSHERS (MEDIUM EASY)

MAKE A NEW TOUCHPLATE SWITCH

A touchplate is a type of switch that is activated by a character standing on it. This tutorial can be used as a guide for making other kinds of switches as well. Switches already in the project include the BP_Switch_Toggle (a standard light switch the character "operates") and BP_Switch_Button (a big button to push the character "operates"). These can be studied for further information.

1. **Setup.** Open (a copy of) the ClassicFPS project. In the Content Browser, navigate to the top-level Content folder, and ensure there is a folder called Tutorials, and in the Tutorials folder, create a new folder called Touchplate.

2. **Create a touchplate mesh.** This can be done in Blender, 3DS Max, Maya, or some other 3D content-generation package. Or you can use the SM_Touchplate asset already in the Content/Solutions/Tutorials/Touchplate folder. For this tutorial, I shall use the free Blender.

   A. Open Blender and delete the default Cube Mesh

   B. Shift-A to add mesh, select Mesh→Cylinder.

C. In the Add Cylinder properties panel (by default, lower left side), ensure the following properties:

   i. Vertices: 32

   ii. Radius: 1

   iii. Depth: .1

   iv. Cap Fill Type: Ngon

   v. Generate UVs: Yes

   vi. Align to View: No

   vii. Location: x=0.000, y=0.000, z=0.000

   viii. Rotation: x=0°, y=0°, z=0°

D. Ctrl-S to save. Choose an appropriate directory and filename (such as "Touchplate.blend"), and save it.

E. Make sure the touchplate mesh is selected (right-click to select), and nothing else.

F. In the top menubar, select File→Export→FBX (.fbx)

G. In the Export FBX properties panel (by default, lower left side), ensure the following properties under the given tabs:

   i. Main Tab:

      a. Version: FBX 7.4 binary

b. Scale: 1.00

c. Apply Scale: All Local

d. Forward: -Z Forward

e. Up: Y Up

f. Selected options: all of them (Empty, Camera, Lamp, Armature, Mesh, Other)

g. !EXPERIMENTAL! Apply Transform: No

h. Custom Properties: No

i. Path Mode: Auto

j. Batch Mode: Off

ii. Geometry Tab

a. Apply Modifiers: Yes

b. Use Modifiers Render Setting: Yes

c. Smoothing: Face

d. Loose Edges: No

e. Tangent Space: No

iii. Armature Tab

a. Only Deform Bones: No

b. Add Leaf Bones: No

c. Primary Bone Axis: Y Axis

d. Secondary Bone Axis: X Axis

e. Armature Node Type: Null

iv. Animation Tab

a. Baked Animation: Yes

b. Key All Bones: Yes

c. NLA Strips: Yes

d. D: All Actions: Yes

e. Force Start/End Keying: Yes

f. Sampling Rate: 1.00

g. Simplify: 1.00

H. Navigate to your Unreal project, then Content directory, then Tutorials directory, then Touchplate directory.

I. Set the name of the file to save to be SM_Touchplate.fbx

J. Click the Export FBX button.

K. Go to UE4 Editor.

L. You should see a popup; select "Import" (if not, click the Import button in the content browser, navigate to where you saved the FBX file, and import it).

M. Now you see an FBX Import Options panel. Ensure the following properties:

   i. Skeletal Mesh: No

   ii. Auto Generate Collision: Yes

   iii. Leave advanced properties at their defaults

   iv. Transform: leave default all zero except scale is 1.0

   v. Miscellaneous: Convert Scene yes, Force Front X Axis No, Convert Scene Unit No, advanced properties, leave defaults

   vi. LOD Settings: Auto Compute LOD Distance Yes

   vii. Materials: Search Location: Local, Import Materials: no, Import Textures: no, leave advanced options as defaults

N. Click Import All.

O. Your touchplate mesh is now in the Content Browser under Content/Tutorials/Touchplate.

P. Optional: give the touchplate a material of some kind (such aSs MI_Rubber_Blue2).

Q. Close the static mesh editor window.

3. Create a Touchplate Blueprint.

A. In the Content Browser in the UE4 Editor, navigate to the Content/Tutorials/Touchplate folder, the same folder where the SM_Touchplate mesh asset is.

B. Right click, then select Blueprint Class. A popup appears titled "Pick Parent Class"

C. If necessary, click the arrow next to "All Classes" to show all classes. In the search bar, begin typing BP_Switch_Base. You do not need to type it all; the BP_Switch_Base class will appear in the hierarchy below the search box.

D. Select BP_Switch_Base (because of a bug, you might have to select it twice to get it highlighted). Then click the green Select button at the bottom of the popup.

E. A new blueprint with the name "NewBlueprint" will appear in the content browser. While the name is still highlighted (if it is not, clicking the asset and hitting F2 will let you rename it), change its name to BP_Touchplate.

F. Double click the new BP_Touchplate asset to open the blueprint editor for it.

G. Make sure the Viewport tab is active (you will see a white sphere in the viewport).

H. To the left (by default), in the Components panel, click the green Add Component button. A dropdown menu will appear

I. Start typing "Static Mesh", and select the Static Mesh option (not either of the two instanced static mesh options, just plain "Static Mesh"). Accept the entry. Hit ENTER to keep the name "StaticMesh" without renaming it.

J. With the StaticMesh component still highlighted, find the Static Mesh section of the Details panel (by default, on the right).

K. Click the box with the down arrow on it and start typing "SM_Touchplate", and when it comes up, select SM_Touchplate for the static mesh. The newly-created Touchplate mesh will appear in the viewport.

L. Find the Collision section in the Details tab. Ensure Generate Overlap Events is checked, and that Collision Presets is set to Overlap All Dynami9c

M. Open the Event Graph tab. If the tab is not there, you can open it by double-clicking the Event Graph item under Graphs in the My Blueprint panel (by default, to the left below the Components panel; if the My Blueprint panel is not there, it can be opened from the menu bar, Window menu). There is a bug in some versions of UE4 so if clicking the EventGraph item doesn't work, right-click and select "Open in New Tab".

N. Highlight the Static Mesh component in the Components tab.

O. To the right (by default), in the Details tab, scroll down, and find a green button with a plus sign in it labeled On Component Hit. Click it. An On Component Begin Overlap (StaticMesh) node appears in the Event Graph blueprint.

P. On the On Component Begin Overlap event node, click and drag rightward from the right-pointing white-outlined arrow to place a new node. When you let go of the mouse button, an Executable Actions popup will appear.

Q. Type "Toggle Action" in the search bar, and when it appears, click on Toggle Action. A new node should appear called Toggle Action, connected to the On Component Begin Overlap node by a white line. (This represents an execution path from the event node to the toggle action node, so

that whenever the touchplate is hit, the toggle action event will be thrown. In the Switch Base, this toggle action just toggles the switch between on and off).

R. Compile the blueprint (Click the compile button in the toolbar), then save the blueprint (click the Save button in the toolbar). Close the blueprint.

4. Place the Touchplate in the Level and Wire it up to something.

A. Create a new level: select File→New Level, and click Default. Use Ctrl-S to save it, and give it the name Touchplate_Test, and save it in the Content/Tutorials/Touchplate directory.

B. Ensure the Content Browser is in Content/Tutorials/Touchplate where you can see the BP_Touchplate and the SM_Touchplate assets.

C. Drag the BP_Touchplate asset into the scene in the viewport, and set it on the default platform somewhere. Currently the touchplate does not do anything because it is not connected to anything controllable.

D. Navigate to Content/ClassicFPS/Environment/Props/Lights in the Content Browser, and find BP_Lamp_Ceiling. Drag this asset into the scene and move it somewhere reasonable (lift it up above the platform by selecting the viewport, clicking W to show the Move widget, and clicking and dragging the blue arrow on the Move widget) to make it visible.

E. Click the Touchplate actor in the scene to select it.

F. In the details panel, in the State section, are a lot of inputs. We shall only use a few of them. Click the white Plus sign to the right of Controlled Objects to add a new array element. Element 0, set to the value None, appears

G. Click the box containing None and the down-pointing triangle to see a list of assets in the current scene. Find and select the BP_Lamp_Ceiling asset. Thus, this touchplate will control the lamp we just placed in the level.

H. Click the right-pointing right triangle next to On Enter State Actions. Currently the action for state 0 is None. Change that to "Off", so when the touchplate is in state 0, the light will be turned off.

I. Click the white Plus sign to the right of On Enter State Actions to add another element to the array. A new element 1 will appear, with value None.

J. Change the None to "On". Thus, when the touchplate enters state 1, the light will be turned on.

K. Note the touchplate's current state is 0. Thus, it starts "Off". Let us make sure the light is off so the states are sync'd when the level is played. Click the light asset in the scene. In the controllable section, click the right-pointing white arrow next to Light. Change the Light is On checkbox to "Unchecked". In the scene, the light will turn off. Now the light and the touchplate are sync'd, both in the "Off" state.

L. Test by clicking the Play button in the toobar. When you walk on the touchplate, the light should turn on. When you walk away and walk back onto the touchplate, the light should turn off.

M. Type CTRL-S to save the level.

N.

NEW WEAPON CREATION

The following tasks need to be performed to create a new weapon.

- Suppose {NAME} is the weapon's name.
- Create a folder under {ProjectDirectory}/Content/ClassicFPS/Pickups/Weapons called {NAME} (you may have to modify the name, such as replacing spaces with underscores or using camelcase or pascal case, to make it a valid folder name). All weapon assets will go here (or in subdirectories).
- Use a 3D digital content creation tool (such as Blender) to create a skeletal mesh and import it into Unreal Editor (be sure to create a skeleton for it). It is recommended you use the Universal Gun Base as a guide for easy fitting to the default character Idle Rifle Hip pose. Give it the name SK_{NAME} or something similar. (Blender Tip: make sure your armature is not called "Armature" or there will be scaling bugs for some reason).
- Open the SK_{NAME} mesh in UE4 and apply appropriate material(s) to it.
- Open the skeleton and add a socket called "Muzzle" to an appropriate bone. Move it to an appropriate location, and rotate it so the X axis points forward and the Y axis points right (from the gun holder's point of view), and the Z axis points up. If it is a projectile weapon, projectiles will be spawned here, so make sure it won't collide with the gun.
- Make sure the skeleton has a physics asset (single convex hull usually works well), and make sure the mesh uses it.
- In the new folder you created, create a new blueprint whose parent blueprint is one of the weapon base blueprints in the directory

{ProjectDirectory}/Content/ClassicFPS/Pickups/Weapons. Most likely you do not want to use the simple BP_Weapon_Base as the parent, because you would have to reinvent the wheel a bit to get it working. Use one of the others.

- Name this new blueprint BP_{NAME} (again the name might need to be modified to make it a valid blueprint name).
- Copy the data table from the folder {ProjectDirectory}/Content/ClassicFPS/Pickups/Weapons to the newly created folder. The data table is called DefaultWeaponProperties. Rename the copy to {Name}Properties or something similar.
- In the BP_{NAME} blueprint, change the icon if you wish (in the Actor section of the Details panel in the Class Defaults). You might select from an existing icon, which has the string "Icon" in its name for easy searching.
- In the BP_{NAME} Blueprint, change the crosshair if you wish (in the Actor|Pickup|Weapon section of the details panel in the Class Defaults). You might select from an existing crosshair image, which has the string "Crosshair" in its name for easy searching.
- In the BP_{NAME} Blueprint, change the Pickup Mesh, First Person Mesh, and Third Person Mesh skeletal mesh values to the skeletal mesh you created (you must use a skeletal mesh weapon; it must have a socket at the appropriate location called "Muzzle"). Optionally, the pickup, first person, and third person meshes can be different (e.g. first person mesh very detailed with ammo counters, etc.)

- In the {NAME}Properties data table, adjust values for fields as needed. For all weapons, you likely want to change the name, possibly the UI colors and additional tags, current ammo (the ammo it starts with), ammo capacity, and Time Between Shots. Other properties will change depending on what the weapon type is. For example, the Hitscan requires Damage Per Shot and Range to be set. Currently Crosshair and Icon are not set from this table, but are set directly in the blueprint.

- In the BP_{NAME} Blueprint, change the Data Table to Use variable to point to your newly-created data table.

- Note that if you set these properties in the blueprint, they will be overridden by the data table! Change the properties in the data table instead. Note some things like the crosshair and icon are still selected in the blueprint; for those, consider the data table entries to be "documentation".

- To make the weapon useable, it needs to be added to the ClassicFPS_GameMode: open the {ProjectDirectory}/Content/ClassicFPS/Gameplay/ClassicFPS_GameMode blueprint. Under MyBlueprint, select the Weapon Slots array variable. The variable is now shown in the Details panel.

- Add another entry to the array, and select your BP_{NAME} weapon as its value. Then rearrange the array at will: the 0^{th} element will be the weapon slot 1, the 1^{st} element the weapon slot 2, and so on that are mapped (by default) to the keys 1, 2, 3, and so on.

ENDNOTES