Classic FPS Game Project Design Document

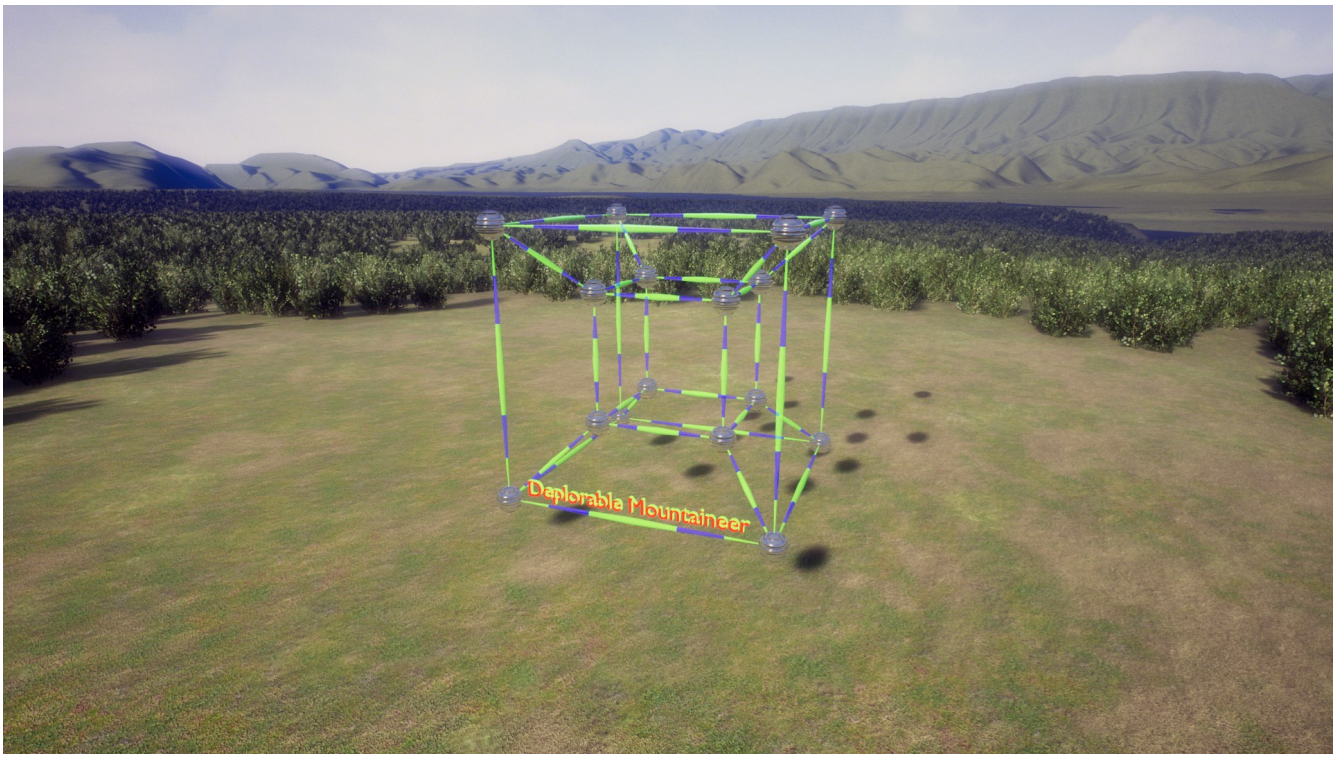


Table of Contents

Game Information

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

Name: Classic FPS Version 0.01

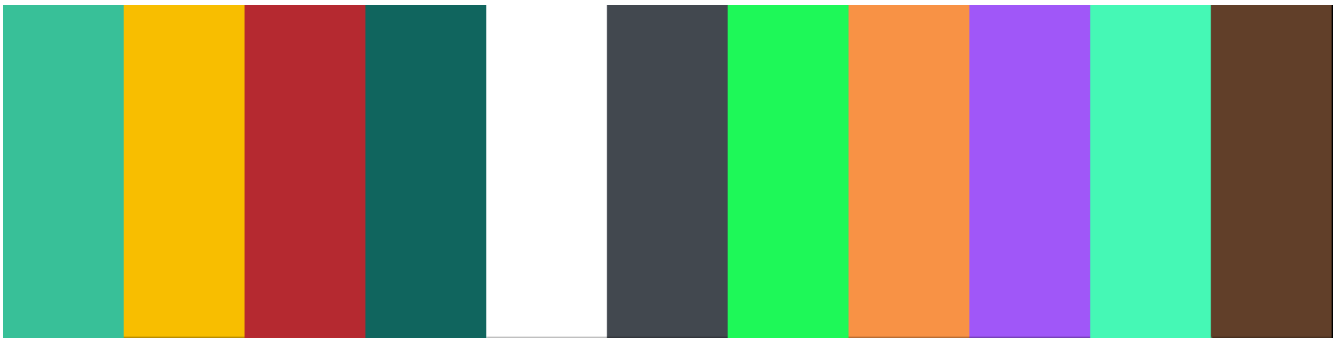
Description: Classic first person shooter project for Unreal Engine 4 in the style of the original single-player Doom, Unreal, Quake, and so on.

Company: Deplorable Mountaineer

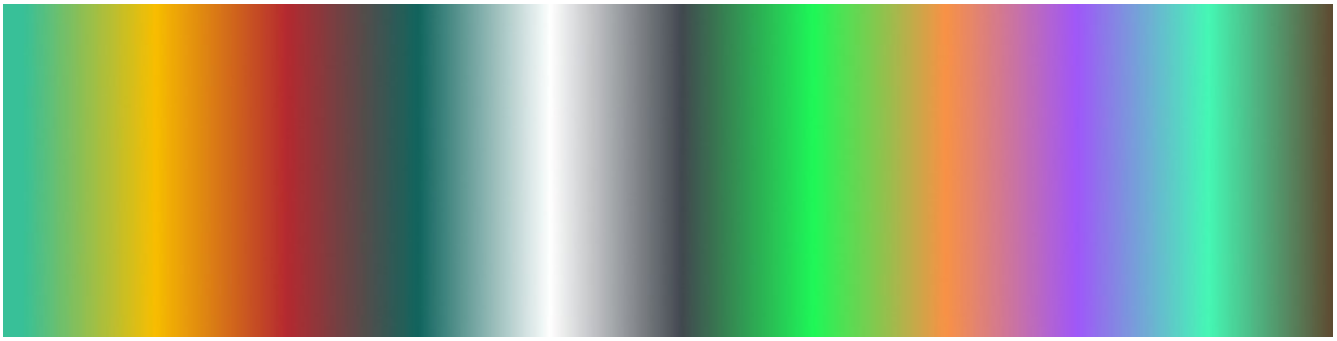
Website: https://deplorablemountain.wixsite.com/games

Contact: Deplorable.Mountaineer@gmail.com

Palette Colors: #38c098, #F8BE00, #B52930, #10655E, #ffffff, #42484f, #1EF858, #F89245, #A057F8, #45F8B5, #613F29



Palette Gradient:



Folder Structure

Under Content:

```
├──AnimStarterPack
│   └──UE4_Mannequin
│       ├──Materials
```

```
│   │   └── MaterialLayers
│   ├── Mesh
│   └── Textures
├── ClassicFPS
│   ├── Base
│   │   ├── Blueprints
│   │   ├── Icons
│   │   ├── MaterialMasters
│   │   │   ├── Data
│   │   │   └── Functions
│   │   └── Physmats
│   │       └── Data
│   ├── Characters
│   │   ├── Animations
│   │   │   ├── AnimationSequences
│   │   │   ├── Blueprints
│   │   │   └── FirstPerson
│   │   ├── Data
│   │   ├── Materials
│   │   │   └── Textures
│   │   ├── NPC
│   │   │   └── Data
│   │   └── Player
│   │       └── Data
│   ├── Dev
│   ├── Effects
│   │   ├── Controllable
│   │   ├── Materials
│   │   │   ├── Ceramic
│   │   │   ├── Data
│   │   │   ├── Decals
│   │   │   │   └── Textures
│   │   │   ├── Fabric
│   │   │   │   └── Textures
│   │   │   ├── FloorCeiling
│   │   │   │   └── Textures
│   │   │   ├── FluidAndIce
│   │   │   ├── Glass
│   │   │   │   └── Textures
│   │   │   ├── Glow
│   │   │   ├── Ground
│   │   │   │   └── Textures
│   │   │   ├── Metal
│   │   │   │   ├── Painted
│   │   │   │   │   └── Textures
│   │   │   │   └── Textures
│   │   │   ├── Paper
│   │   │   │   └── Textures
│   │   │   ├── Plastic
│   │   │   ├── Rock
│   │   │   │   └── Textures
│   │   │   ├── Rubber
│   │   │   │   └── Textures
│   │   │   ├── Special
│   │   │   ├── Textures
│   │   │   ├── Wall
```

```
│   │   │   │   └──Textures
│   │   │   └──Wood
│   │   │       └──Textures
│   │   └──Particles
│   │       └──Materials
│   ├──Environment
│   │   ├──Background
│   │   ├──Buildings
│   │   │   ├──Commercial
│   │   │   ├──Fixtures
│   │   │   ├──Furniture
│   │   │   ├──Pipes
│   │   │   └──Residential
│   │   ├──Foliage
│   │   ├──Landscape
│   │   │   └──Rocks
│   │   ├──Props
│   │   │   ├──Lights
│   │   │   │   └──Meshes
│   │   │   ├──Meshes
│   │   │   └──Shapes
│   │   ├──Sky
│   │   └──Water
│   ├──Gameplay
│   ├──Maps
│   │   ├──Episode_01
│   │   ├──Episode_02
│   │   ├──Episode_03
│   │   └──TestMaps
│   ├──Pickups
│   │   ├──Health
│   │   ├──Keys
│   │   ├──Powerups
│   │   │   └──Permanent
│   │   └──Weapons
│   │       ├──Ammo
│   │       ├──Materials
│   │       │   └──Textures
│   │       │       └──Crosshairs
│   │       └──Rifle
│   ├──PostProcess
│   ├──Sound
│   ├──UI
│   │   ├──HUD
│   │   │   └──Materials
│   │   ├──MainMenu
│   │   │   └──Materials
│   │   ├──Monitor
│   │   └──PauseMenu
│   └──Vehicles
├──Collections
├──Developers
│   └──tdvance
│       └──Collections
├──FirstPerson
│   ├──Animations
```

```
│   ├──Audio
│   ├──Character
│   │   ├──Materials
│   │   │   └──MaterialLayers
│   │   ├──Mesh
│   │   └──Textures
│   ├──FPWeapon
│   │   ├──Materials
│   │   │   └──MaterialLayers
│   │   ├──Mesh
│   │   └──Textures
│   ├──Meshes
│   └──Textures
├──FirstPersonBP
│   ├──Blueprints
│   └──Maps
├──Geometry
│   └──Meshes
├──Mannequin
│   ├──Animations
│   └──Character
│       ├──Materials
│       │   └──MaterialLayers
│       ├──Mesh
│       └──Textures
├──ThirdPerson
│   └──Meshes
└──ThirdPersonBP
    ├──Blueprints
    └──Maps
```

Assets

Material Bases

PBR Settings

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

Material	Diffuse	Metallic	Specular	Roughness	Emissive	Opacity	Subsurface	IOR
Aluminum	.913,.921,.925	1	0.5	0.25		1	0,0,0	1
Asphalt	#5E5F64	0	0.5	0.6		1		1
Brick	#E7D2D2	0	0.5	0.7		1		1
Bronze	#fcaf6c	1	0.5	0.25		1	0,0,0	1

Canvas	#8B8D98	0	0.5	0.7		1		1
Ceramic	#C7B7A4	0.1	0.5	0.25		1		1
Chromium	.55,.556,.554	1	0.5	0.25		1	0,0,0	1
Coal	#42413E	0	0.5	0.5		1		1
Cobalt	.662,.655,634	1	0.5	0.25		1	0,0,0	1
Concrete	#84837B	0	0.5	0.7		1		1
Copper	.955,.637,.538	1	0.5	0.25		1	0,0,0	1
Diamond	0,0,0	0.1	0.5	0.25		0.5	0,0,0	2.5
Dirt	#150700	0	0.5	0.7		1		1
Energy	.1,.3,.6	0.1	0.5	0		0.5		1.2
Fabric	.5,.5,.5	0	0.5	0.5		0.9		1
Foam	.7,.7,.5	0	0.5	0.4		0.8		1
Glass	0,0,0	0.1	0.5	0.25		0.5	0,0,0	1.4
Glass Broken	0,0,0	0.1	0.5	0.4		0.8		1.4
Gold	.1,.766,.336	1	0.5	0.25		1	0,0,0	1
Granite	#A28871	0	0.5	0.7		1		1
Grass	#A1BE32	0	0.5	0.6		1		1
Gravel	#69716D	0	0.5	0.7		1		1
Ice	.8,.8,.8	0	0.224	0.25		0.8	0,0,0	1.3
Iron	.56,.57,.58	1	0.5	0.25		1	0,0,0	1

Lava	black body	0	0.5	0.6		1	black body, different pattern’	1
Lead	#cacaca	1	0.5	0.25		1	0,0,0	1
Leather	#9D5C34	0	0.5	0.5		1		1
Leaves	#55592D	0	0.5	0.7		1		1
Limestone	#CDB189	0	0.5	0.7		1		1
Mercury	#c9c7c7	1	0.5	0.05		1	0,0,0	1
Milk	.8,.8,.8	0	0.277	0.25		1		1
Mud	#9E8871	0	0.5	0.5		1		1
Nickel	.66,.609,.526	1	0.5	0.25		1	0,0,0	1
Oil	#B96D2F	0	0.5	0.25		0.9		1.5
Painted Metal	#964A2E	0.5	0.5	0.4		1		1
Paper	.7,.7,.65	0	0.5	0.4		1		1
Pewter	#beb4a8	1	0.5	0.25		1	0,0,0	1
Plaster	#8F8C85	0	0.5	0.86		1		1
Plastic Rough	#6A7F8E	0	0.5	0.7		1		1
Plastic Smooth	#385180	0	0.5	0.54		1		1
Platinum	.672,.637,.585	1	0.5	0.25		1	0,0,0	1
Quartz	0,0,0	0	0.57	0.25		0.7	0,0,0	1.5
Rock	#AA6060	0	0.5	0.7		1		1
Rough Steel	#6D6662	1	0.5	0.5		1	0,0,0	1

Rubber	#42413D	0	0.5	0.9		1	0,0,0	1
Ruby	1,0,0	0.1	0.5	0.25		0.5	0,0,0	1.8
Rust	#773115	0	0.5	0.9		1		1
Sand	#DDB97F	0	0.5	0.7		1		1
Satin	#AF4433	0	0.5	0.25		1		1
Silver	.972,.96,.915	1	0.5	0.25		1	0,0,0	1
Skin	Varies	0	0.35	0.5		1		1
Slate	#5F6B79	0	0.5	0.7		1		1
Slime	.1,.8,.3	0	0.5	0.2		0.5		1
Snow	.8,.8,.8	0	0.5	0.25		0.9		1
Tin	#e4e4e4	1	0.5	0.25		1	0,0,0	1
Titanium	.541,.497,.449	1	0.5	0.25		1	0,0,0	1
Water	0,0,.1	0	0.255	0.25		0.5	0,0,0	1.3
Wood	#AA9984	0	0.5	0.68		1		1

Material Masters

Base, Translucent_Expensive, Mask

Parameter Names

Material Parameters				
Texture	Multiplier	Static Switch	Bias	Notes
Diffuse	Tint	Use Diffuse Texture	Bias	Alpha is opacity or mask (or subsurface alpha)
Metallic Mask	Metallic Multiplier	Use Metallic Mask	Metallic	Use blue channel
Specular Mask	Specular Multiplier	Use Specular Mask	Specular	Use blue channel

Roughness Mask	Roughness Multiplier	Use Roughness Mask	Roughness	Use Blue Channel
Emissive	Emissive Tint	Use Emissive Texture	Emissive Bias	
Normal		Use Normal Texture		If not using texture, #0000ff constant used instead
Bump	Bump Amount	Use Bump		Use Blue Channel
Subsurface	Subsurface Tint	Use Subsurface Texture	Subsurface Bias	Use Blue Channel
AO	AO Multiplier	Use AO		
Fresnel Mask	Fresnel Multiplier	Use Fresnel Mask	Fresnel	Use Blue Channel
IOR Mask	IOR Multiplier	Use IOR Mask	IOR	Use Blue Channel
Clearcoat	Clearcoat Tint	Use Clearcoat Texture	Clearcoat Bias	
Clearcoat Roughness Mask	Clearcoat Roughness Multiplier	Use Clearcoat Roughness Mask	Clearcoat Roughness	
	Emissive Lerp Alpha			for black-light, night vision, etc. effects
	X Scale		X Shift	For correcting the UV
	Y Scale		Y Shift	
		Apply Object Scale XY		
		Apply Object Scale XZ		
		Apply Object Scale YZ		
		Apply Object Scale YX		
		Apply Object Scale ZX		
		Apply Object Scale ZY		
		Swap U and V		

		Invert Metallic		For correcting textures
		Invert Specular		
		Invert Roughness		
Mask				For RGB 3-channel mask; Diffuse_R, Diffuse_G, etc. used
Mask8				For RGBCMYKW 8-channel mask
Macro	Macro Scale	Use Macro	Macro Rough Bias	
Micro	Micro Scale	Use Micro	Micro Rough Bias	
		Use Macro Diffuse Multiplier		
		Use Micro Diffuse Multiplier		
		Use Macro Rough Multiplier		
		Use Micro Rough Multiplier		

Material Functions and Layers

MF_MaskLerpRGB

Textures

Crosshairs: Cross, X, Dot, Triangle, Trefoil, Box, Circle, Rangefinder

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: Default, Actor, Character, Weapon, Carryable

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

Painted metal, colored glass, patterned cloth, etc.

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

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

Hazard patterns

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

Diamond Plate, Duct, Planks, Parquet

Particles

Explosions, Hit effect, sparks (fire, electric, hammered metal), Fires, Beams, Projectile particles, Splash from water, slime, lava, Snow, Steam, Smoke, Lightning, Shrapnel, Gibs, Dust, Contrail, Light halo,

Actor Base

Properties

Name

Unique Name (just use name-in-world)

Icon

Colors

Additional Tags (automatically populates list of actor tags)

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

Close

Open Reverse

Close Reverse

Lock

Unlock

Operate

- If operated from -X direction:
  - if open, close
  - if open reverse, close reverse
  - if closed and unlocked, 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

Close Speed

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

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.

Architectural Parameters

Residential	
Room Height	300
Door Height	200
Door Width	100
Window Height	150
Window Width	100
Wide Window Width	150
Garage Door Height	225
Garage Door Width	375
Interior Wall Thickness	20
Exterior Wall Thickness	20
Commercial	
Room Height	400
Door Height Small	200

Door Width Small	100
Door Height Medium	300
Door Width Medium	200
Door Height Large	400
Door Width Large	300
Garage Door Height	750
Garage Door Width	1000
Interior Wall Thickness	20
Exterior Wall Thickness	100
Roads	
Lane Width	400
sidewalk minimum width	200
Parking space length	600
Parking space width	300

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

Barrels, Crates, including exploding barrel, Boxes, bags

Bricks, Cement blocks, Rocks, Boulders

Foliage

Cones, Saw horses, Signs, Manhole covers

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

Grates, vents,

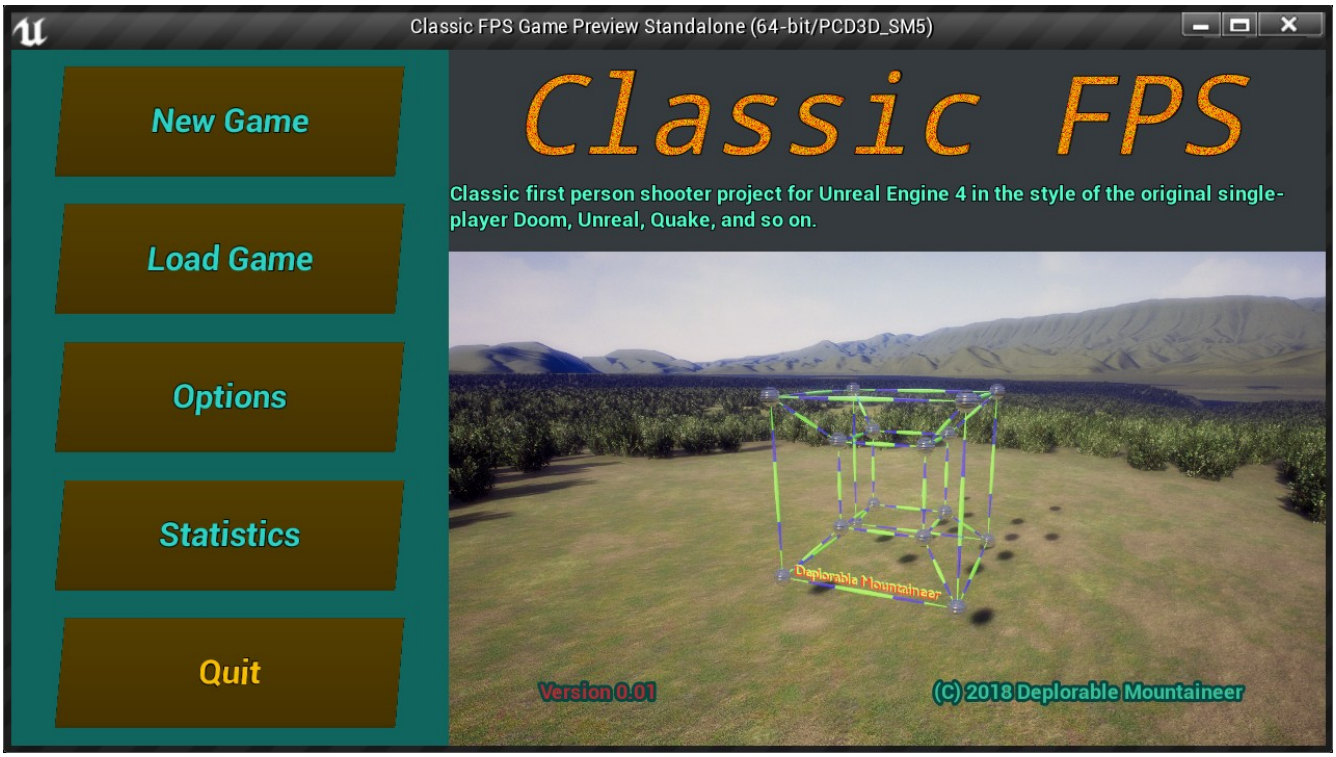
Gameplay Base

Gameplay States

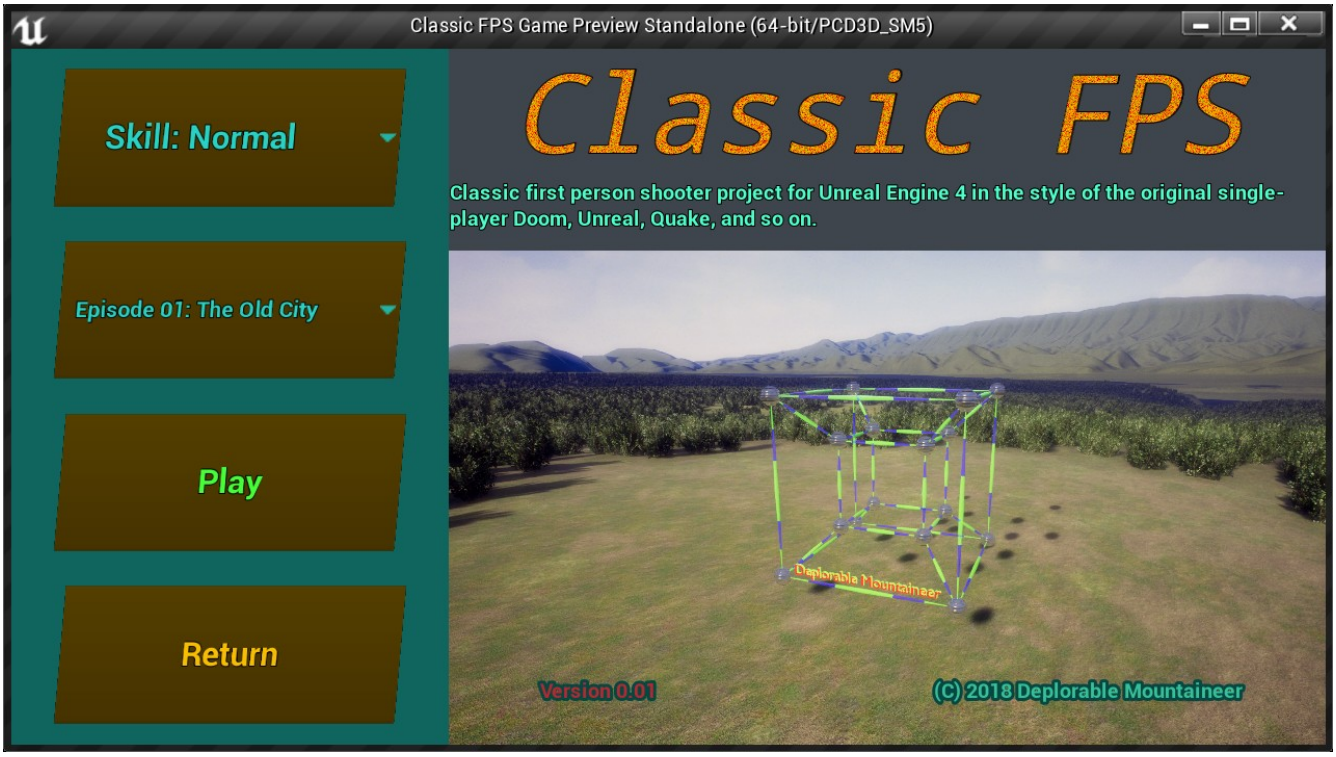
Main Menu

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

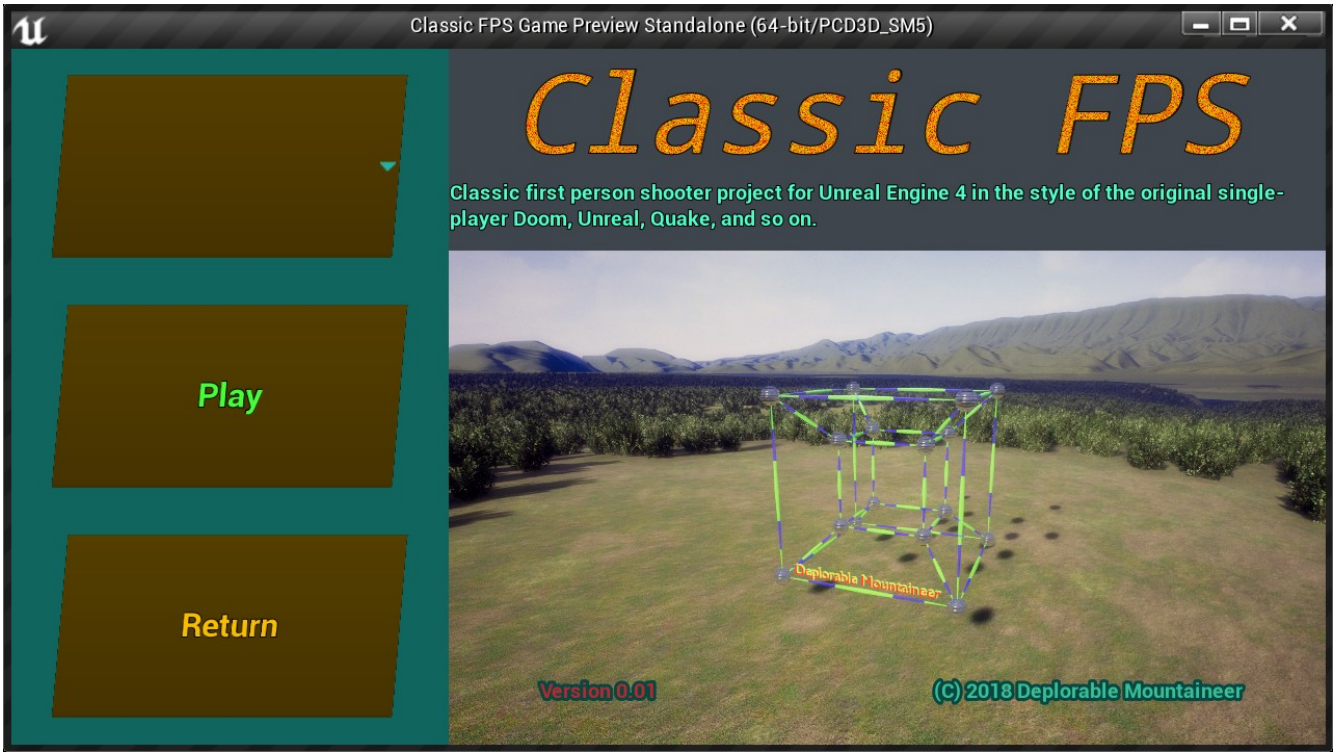
The main menu is brought up by the BeginPlay event of the Level Blueprint of the Entry map. This same level blueprint has properties UI Colors and UI Text for changing various menu properties for all menus.



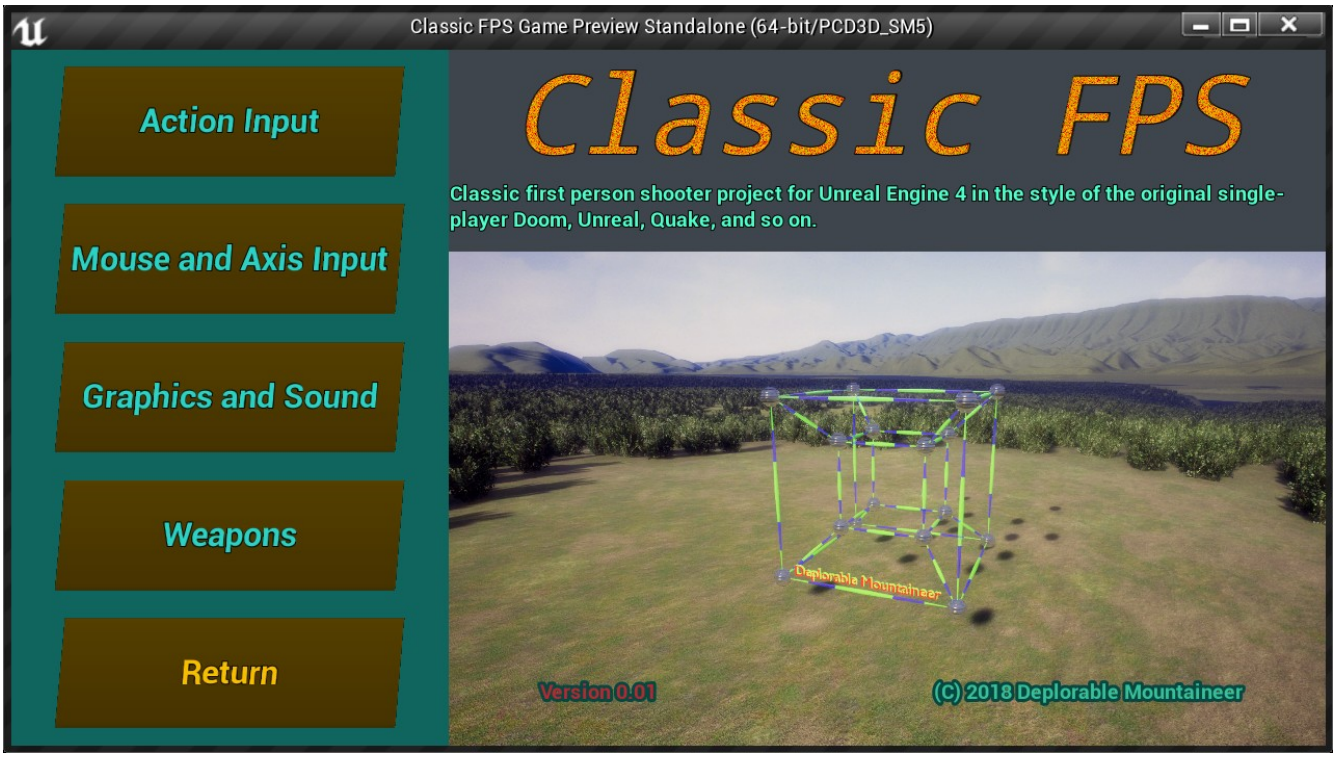
New Game: Select Skill, Select Episode, Play, Return



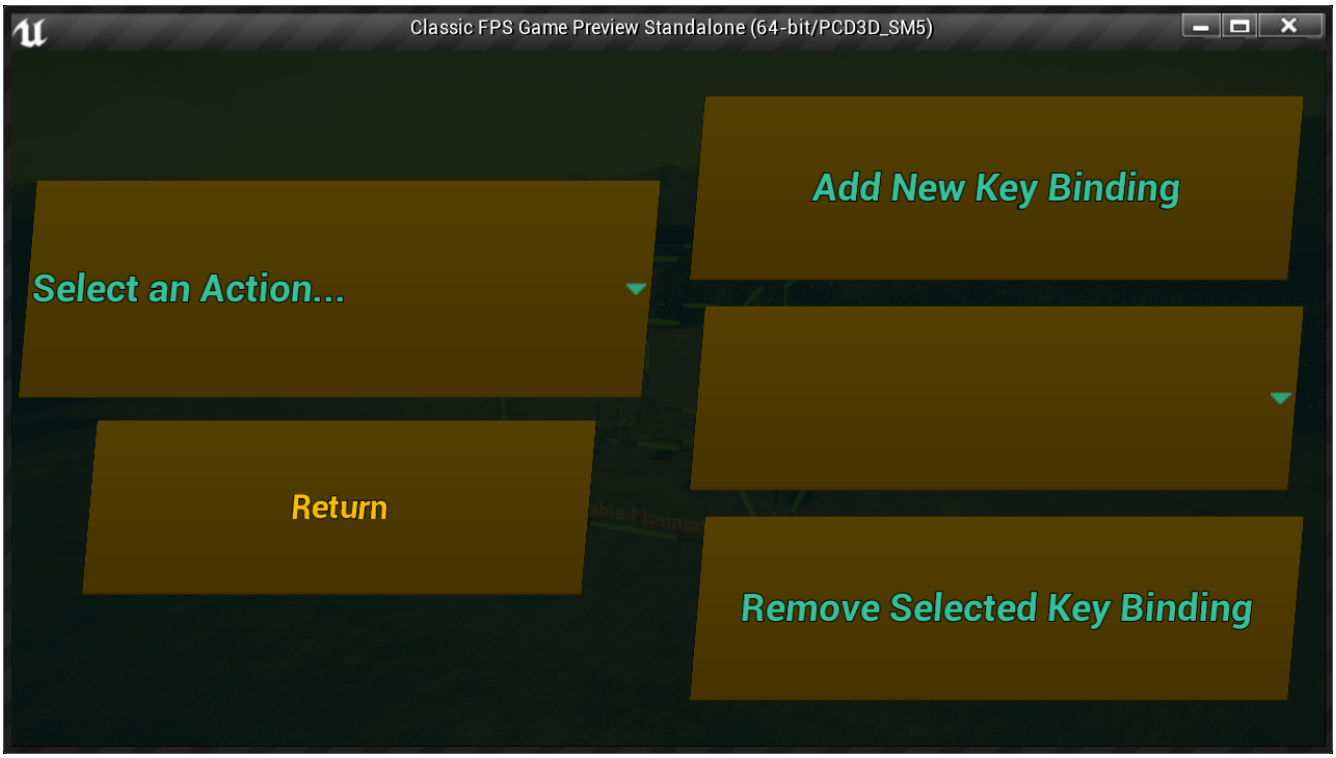
Load Game: Select Game, Play, Return



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

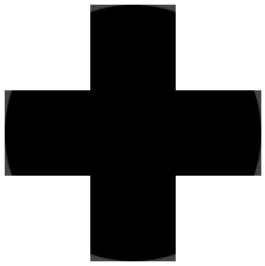
Pause Menu

HUD

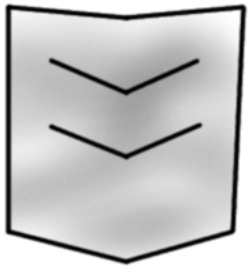
Ammo on bottom right, change color when low, with flash when it turns low and audible noise;

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.

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.

ClassicFPS Game State

- Current level
- Levels Completed
- Strength Achieved

ClassicFPS Player State

- Skill Level
- Name

Gameplay Effects

Hurt screen

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

Property	Float Value	String Value	Vector Value X	Vector Value Y	Vector Value Z
Actor Properties					
Additional Tag 1					
Additional Tag 2					
Additional Tag 3					
Color 1		Salem	0.039546	0.527115	0.313989

Color 2		Gold Drop	0.941177	0.513726	0
Color 3		Tall Poppy	0.462077	0.022174	0.029557
Color 4		Eden	0.005182	0.130136	0.111932
Color 5		White	1	1	1
Color 6		Bunker	0.05098	0.062745	0.078431
Icon	0	T_Man_Icon_M			
Name		Character			
AI Characteristics					
Accuracy	0.5				
Aggressiveness	0.5				
Alertness	0.5				
Jumpiness	0.5				
Map Awareness	0.5				
Reaction time	0.5				
Tactics	0.5				
Character Movement					
Air Control	0.05				
Air Control Boost Multiplier	2				

Air control Boost Velocity Threshold	25				
Base Jump Velocity	400				
Base Running Jump Velocity	600				
Brake Deceleration Falling	0				
Default Ground Friction	8				
Falling Lateral Friction	0				
Max Crouch Walk Speed	300				
Max Fly Speed	600				
Max Step Height	45				
Max Swim Speed	300				
Max Walk Speed	600				
Walkable Floor Angle	45				
Other Character Properties					
Auto-Kill Fall Time	10				
Base Eye Height	64				
Camera Boom Location			-30.846336	17.973167	44.23

Carry Distance	100				
Character Height	192				
Character Width	84				
Crouch Time	0.2				
Crouched Character Half Height	40				
Crouched Eye Height	32				
Gravity Scale	1				
Mass	100				
Max FOV	120				
Max survivable fall time	2				
Min FOV	30				
Min Time for Fall Damage	0.4				
Min Velocity for Falling Counter	700				
Minimum Strength	-2				
Num Zoom Levels	4				
Operate Range	200				
Spring Arm Multiplier per Increment	1.1				
Strength Adjust	0				
Strength Multiplier per Increment	1.1				
Third Person Camera Arm Length	300				

Mesh Properties					
First Person Anim Class		ABP_Arms			
First Person Arms Location Rel Camera			-0.510596	-4.410996	-155.712738
First Person Arms Rotation Rel Camera			5.287787	1.921141	-19.911278
First Person Gun Socket Bone		middle_01_r			
First Person Gun Socket Location			-2.457152	6.927716	0.000142
First Person Gun Socket Rotation			4.847672	15	175
First Person Mesh		SK_Arms			
First Person Mesh Override Material 1					
First Person Mesh Override Material 2					
Third Person Anim Class		ABP_Character			
Third Person Gun Socket Bone		middle_01_r			
Third Person Gun Socket			-1.342635	3.318012	-1.679417

Location					
Third Person Gun Socket Rotation			-22.681015	12.774652	162.533691
Third Person Mesh		SK_Character			
Third Person Mesh Override Material 1					
Third Person Mesh Override Material 2					
Weapon Order					
Weapon preference 1	1				
Weapon preference 2	2				
Weapon preference 3	3				
Weapon preference 4	4				
Weapon preference 5	5				
Weapon preference 6	6				
Weapon preference 7	7				
Weapon preference 8	8				
Weapon preference 9	9				

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.

			Strength								
		-4	-3	-2	-1	0	1	2	3	4	5
Gravity acceleration per G	-980										
Factor		0.62	0.68	0.75	0.83	0.91	1.00	1.10	1.21	1.33	1.46
Inverse Factor		1.61	1.46	1.33	1.21	1.10	1.00	0.91	0.83	0.75	0.68
Jump Velocity		248.37	273.21	300.53	330.58	363.64	400.00	440.00	484.00	532.40	585.64
Running Jump Velocity		372.55	409.81	450.79	495.87	545.45	600.00	660.00	726.00	798.60	878.46
Walk Speed		372.55	409.81	450.79	495.87	545.45	600.00	660.00	726.00	798.60	878.46
Crouch Walk Speed		186.28	204.90	225.39	247.93	272.73	300.00	330.00	363.00	399.30	439.23
Fly Speed		372.55	409.81	450.79	495.87	545.45	600.00	660.00	726.00	798.60	878.46
Swim Speed		186.28	204.90	225.39	247.93	272.73	300.00	330.00	363.00	399.30	439.23
Jump Time		0.25	0.28	0.31	0.34	0.37	0.41	0.45	0.49	0.54	0.60
Double Jump Time		0.51	0.56	0.61	0.67	0.74	0.82	0.90	0.99	1.09	1.20

Running Jump Time		0.38	0.42	0.46	0.51	0.56	0.61	0.67	0.74	0.81	0.90
Running Double Jump Time		0.76	0.84	0.92	1.01	1.11	1.22	1.35	1.48	1.63	1.79
Jump Height		31.47	38.08	46.08	55.76	67.47	81.63	98.78	119.52	144.62	174.99
Double Jump Height		62.95	76.16	92.16	111.51	134.93	163.27	197.55	239.04	289.23	349.97
Running Jump Height		70.81	85.69	103.68	125.45	151.80	183.67	222.24	268.92	325.39	393.72
Running Double Jump Height		141.63	171.37	207.36	250.90	303.59	367.35	444.49	537.83	650.78	787.44
Time To Reach Kill Velocity		0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71	0.71
Distance to Reach Kill Velocity		250.00	250.00	250.00	250.00	250.00	250.00	250.00	250.00	250.00	250.00
Min Time for Fall Damage		0.25	0.27	0.30	0.33	0.36	0.40	0.44	0.48	0.53	0.59
Max Survivable Fall Time		1.24	1.37	1.50	1.65	1.82	2.00	2.20	2.42	2.66	2.93
Min Height for Fall Damage		454.08	477.82	504.62	534.95	569.34	608.40	652.86	703.59	761.57	828.01
Max Survivable Fall Height		1,874.95	2,120.57	2,408.21	2,745.73	3,142.56	3,610.00	4,161.60	4,813.64	5,585.66	6,501.17

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.

Axis Bindings

Action	Key	Shift	Ctrl	Alt	Cmd	Key2	Shift2	Ctrl2	Alt2	Cmd2
Crouch	C									
Fire	LeftMouseButton									
Jump	SpaceBar									
Toggle First/Third Person View	F7									
Zoom	Z									
Switch to Next Weapon	MouseWheelUp					]				
Switch to Previous Weapon	MouseWheelDown					[				
Switch to Best Weapon	B									
Switch to No Weapon	0									
Toss Current Weapon	T									
Alt Fire	MiddleMouseButton									
Operate	E					F				
Select Weapon 1	1									
Select	2									

Weapon 2										
Select Weapon 3	3									
Select Weapon 4	4									
Select Weapon 5	5									
Select Weapon 6	6									
Select Weapon 7	7									
Select Weapon 8	8									
Select Weapon 9	9									
Toggle Night Vision	N									
Toggle Lift Belt	L									
Toggle Nitro Boost	R									
Pause	Esc					Pause				
Increase Third Person Follow	MouseWheelUp		x			[		x		
Decrease Third Person Follow	MouseWheelDown		x			[		x		

Lunge	V									

Axis Bindings

Axis	Key	Scale	Key2	Scale2	Key3	Scale3	Key4	Scale4	Key5	Scale5
Look Up/Down	Mouse Y	1								
Move Right/Left	A	-1	D	1						
Turn Right/Left	Mouse X	1	Right	1	Left	-1				
Move Forward/Backward	W	1	S	-1	Up	1	Down	-1	RightMouseButton	1
(Dodge by double-tap, or forward with V key)										
Increase/Decrease Third Person Follow	Ctrl-MousewheelUp	1	Ctrl-MousewheelDown	-1						

Character Properties

Property	**Type**
Health	Float
Strength	Float
Inventory	Array of Inventory Item Structs
Current Weapon Slot	Integer

Third Person Camera Arm Length	Float
Is Third Person Mode	Bool
Location	Vector
Rotation	Rotator
Enemies Killed	Set of Unique Names
Secrets Found	Set of Names
Powerups Found	Set of Unique Names
Weapons Found	Set of Unique Names
Ammo Found	Set of Unique Names
Health Found	Set of Unique Names
Keys Found	Set of Names
Falling Counter	Float
FOV	Float
Carried Item	Physics Mesh Property Struct

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 9, meaning there is a hard limit of 9 weapons. 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. 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 adjusted while the character is falling at greater than the minimum velocity, and reset when the player lands and damage is taken.

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 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).

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.

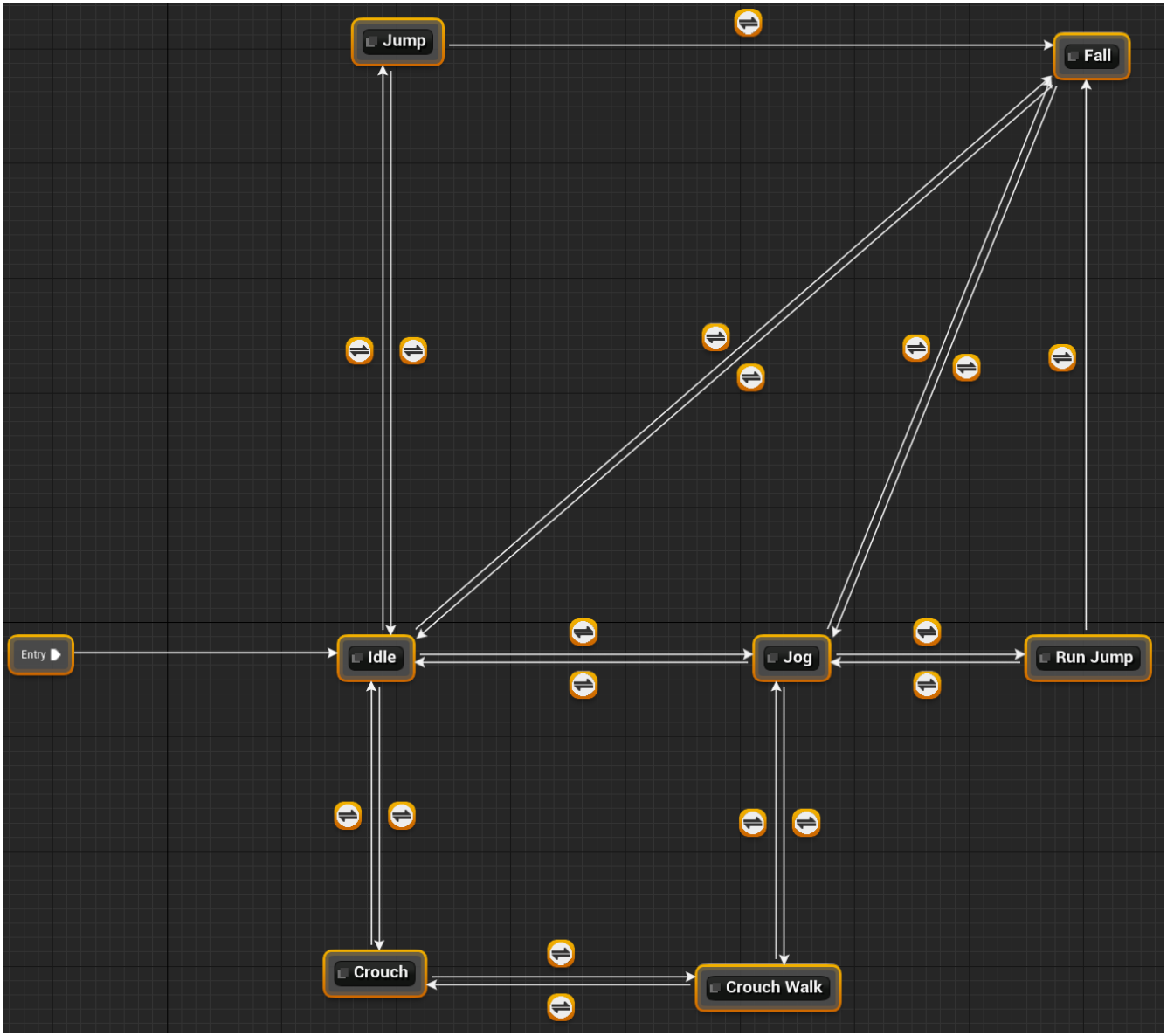
When the Operate action occurs and the character is already carrying something, it drops the item instead of performing the usual Operate procedure.

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.

Simple Character AI

Note that for the AI to work correctly, the player needs to be tagged with Player and all bots with NPC. The Additional Tags are currently set up to do this for the provided player and NPCs.

Bot_Random

Moves in a straight line, turning right when it approaches an obstacle (as viewed straight ahead from its centroid), and at random times, turning in a random direction. Tries to use navmesh if one exists instead of just going in straight lines. If nav movement fails, then moves randomly without the nav mesh. Red.

Bot_Waypoint_Flee_Attack

This bot has four states: waypoint, wander, chase, and flee. It starts in the Waypoint state where, if there are one or more waypoints added to the bot's properties, it tries to go to them in a sequence, 0, 1, 2, to the last and back to 0 again to complete the circuit. It tries nav walking, and if that fails, it tries moving directly toward the waypoint. , turning on meeting an obstacle.

If there are no waypoints, it goes to the Wander state and wanders in a similar way to Bot_Random. If stuck, it attempts randomly jumping and crouching to try to get out of it.

If it is within a specified distance from the player, it will either go to the chase state or the flee state, the former if its health is at least 50, the latter if not. In the chase state, it chases the player (nav walking to player if possible, moving toward player if not, turning at obstacles). In the flee state, it tries to move in the direction away from the player, turning at obstacles. Upon exceeding the specified distance from the player, it reverts to the Waypoint state.

At random times, it changes to a wander state, or if already in a wander state, to a waypoint state, to keep its behavior from being monotonous. Also at random times, it fires the weapon at the player if close enough. White

Bot_Basic

Following are variables the bot maintains. Numbers mentioned may be 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 20 seconds

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 health, 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 the second, 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

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

Powerup Base

Ammo Base

Weapon Base

Property	Float Value
Actor Properties	
Name	
Color 1	
Color 2	
Color 3	
Color 4	
Color 5	
Color 6	
Additional Tag 1	
Additional Tag 2	

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.

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.

Physics Mesh Base

Other Blueprint Objects

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

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.

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

Fans

Sounds

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

Physics

Collision

Note, most values are the UE4 Defaults.

			Collision Response Trace Type						Collision Response Object Type												
Profile	Collision	Object Type	Visibility	Camera	Shootable				WorldStatic	WorldDynamic	Pawn	PhysicsBody	Vehicle	Destructible	Projectile						Description
NoCollision	None	WorldStatic	Ignore	Ignore	Ignore				Ignore	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore						
BlockAll	Both	WorldStatic	Block	Block	Block				Block	Block	Block	Block	Block	Block	Block						
OverlapAll	Query	WorldStatic	Overlap	Overlap	Overlap				Overlap	Overlap	Overlap	Overlap	Overlap	Overlap	Overlap						
BlockAllDynamic	Both	WorldDynamic	Block	Block	Block				Block	Block	Block	Block	Block	Block	Block						
OverlapAllDynamic	Query	WorldDynamic	Overlap	Overlap	Overlap				Overlap	Overlap	Overlap	Overlap	Overlap	Overlap	Overlap						
IgnoreOnlyPawn	Query	WorldDynamic	Block	Block	Block				Block	Block	Ignore	Block	Ignore	Block	Block						
OverlapOnlyPawn	Query	WorldDynamic	Block	Overlap	Overlap				Block	Block	Overlap	Block	Overlap	Block	Block						
Pawn	Both	Pawn	Ignore	Ignore	Block				Block	Block	Block	Block	Block	Block	Overlap						
Spectator	Query	Pawn	Ignore	Ignore	Block				Ignore	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore						
Chara	Quer	Pawn	Ignor	Block	Block				Block	Block	Ignor	Block	Ignor	Block	Overl						

cterMesh	y		e								e		e		ap						
PhysicsActor	Both	PhysicsBody	Block	Block	Block				Block	Block	Block	Block	Block	Block	Block						
Destructible	Both	Destructible	Block	Block	Block				Block	Block	Block	Block	Block	Block	Block						
InvisibleWall	Both	WorldStatic	Ignore	Block	Block				Block	Block	Block	Block	Block	Block	Ignore						
InvisibleWallDynamic	Both	WorldDynamic	Ignore	Block	Block				Block	Block	Block	Block	Block	Block	Ignore						
Trigger	Query	WorldDynamic	Ignore	Ignore	Ignore				Overlap	Overlap	Overlap	Overlap	Overlap	Overlap	Ignore						
Ragdoll	Both	PhysicsBody	Block	Block	Block				Block	Block	Ignore	Block	Block	Block	Overlap						
Vehicle	Both	Vehicle	Block	Block	Block				Block	Block	Block	Block	Block	Block	Overlap						
UI	Query	WorldDynamic	Block	Overlap	Block				Overlap	Overlap	Overlap	Overlap	Overlap	Overlap	Block						
Projectile	Query	Projectile	Ignore	Ignore	Ignore				Block	Block	Overlap	Block	Overlap	Block	Ignore						
PhysicsPickup	Both	PhysicsBody	Block	Block	Block				Block	Block	Overlap	Block	Block	Block	Block						

Physical Surfaces

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

Surface	Hollowness	Resonance	Fiberousness	Brittleness	Dentableness	Friction	Restitution	Density	Roughness	Metallic	Specular
Asphalt	0.1	0.1	0.1	0.1	0.1	0.5	0.9	2.5	0.6	0	0.5
Brick	0.1	0.1	0.1	0.4	0	0.9	0.5	2	0.8	0	0.5
Ceramic	0.2	0.5	0	1	0	0.2	0.8	2	0.1	0.1	0.5
Concrete	0.1	0.1	0.1	0.5	0	0.6	0.5	2	0.7	0	0.5
Dirt	0	0	0	0	0.4	0.6	0.5	3	0.7	0	0.5
Energy	0	0	0	0	0	0	1	0	0	0.1	0.5
Fabric	0	0	1	0	0.1	0.3	0.6	1.5	0.5	0	0.5
Foam	0	0	0	0	0.5	0.7	0.1	0.075	0.4	0	0.5
Glass	0.2	0.7	0.2	1	0	0.5	0.65	2	0.1	0.1	0.5

Glass Broken								2		0.1	0.5
Grass	0	0	0.5	0	0.2	0.2	0.5	3	0.6	0	0.5
Ice	0.1	0.1	0.3	0.5	0	0.03	0.5	0.9	0.2	0	0.224
Lava	0	0	0	0	0	0	0	3	0.6	0	0.5
Leather	0	0	0.1	0	0.5	0.8	0.3	1.5	0.5	0	0.5
Metal Rough	0.3	0.8	0	0	1	0.8	0.6	7	0.7	1	0.5
Metal Smooth	0.3	0.8	0	0	1	0.5	0.6	7	0.4	1	0.5
Mud	0	0	0	0	1	1.5	0	1.5	0.7	0	0.5
Oil	0	0	0	0	0	0	0	0.5	0	0.1	0.5
Paper	0	0	0.4	0	0	0.3	0.3	0.7	0.4	0	0.5
Plaster	0.1	0.1	0.5	0.5	0.1	0.5	0.5	2	0.7	0	0.5
Plastic	0.1	0.1	0.1	0.2	0.5	0.35	0.7	1.2	0.3	0	0.5
Rock Gravel	0.1	0.1	0.1	0.1	0.1	0.6	0.8	3	0.7	0	0.5
Rock Rough	0.1	0.1	0.1	0.4	0.1	0.75	0.8	3	0.7	0	0.5
Rock Smooth	0.1	0.1	0.1	0.4	0.1	0.4	0.8	3	0.7	0	0.5
Rubber	0.2	0.1	0.2	0.1	0.1	1	0.9	1.1	0.7	0	0.5
Sand	0	0	0	0	0.3	0.5	0.1	3	0.7	0	0.5
Slime	0	0	0	0	0	0	0	1	0.2	0	0.5
Snow	0	0	0	0	0.3	0.5	0.1	0.1	0.25	0	0.5
Water	0	0	0	0	0	0	0	1	0	0	0.255
Water Shallow	0.1	0.1	0	0	0	0	0	1	0	0	0.255
Wood	0.6	0.5	0.5	0.2	0.3	0.7	0.6	0.7	0.6	0	0.5

Utility

Default Grid for visualizing UV maps:

Red Orange Yellow Yellow-Green Green Blue-Green Blue Blue-Violet Violet Red-Violet

J0 J1 J2 J3 J4 J5 J6 J7 J8 J9

I0 I1 I2 I3 I4 I5 I6 I7 I8 I9

H0 H1 H2 H3 H4 H5 H6 H7 H8 H9

G0 G1 G2 G3 G4 G5 G6 G7 G8 G9

F0 F1 F2 F3 F4 F5 F6 F7 F8 F9

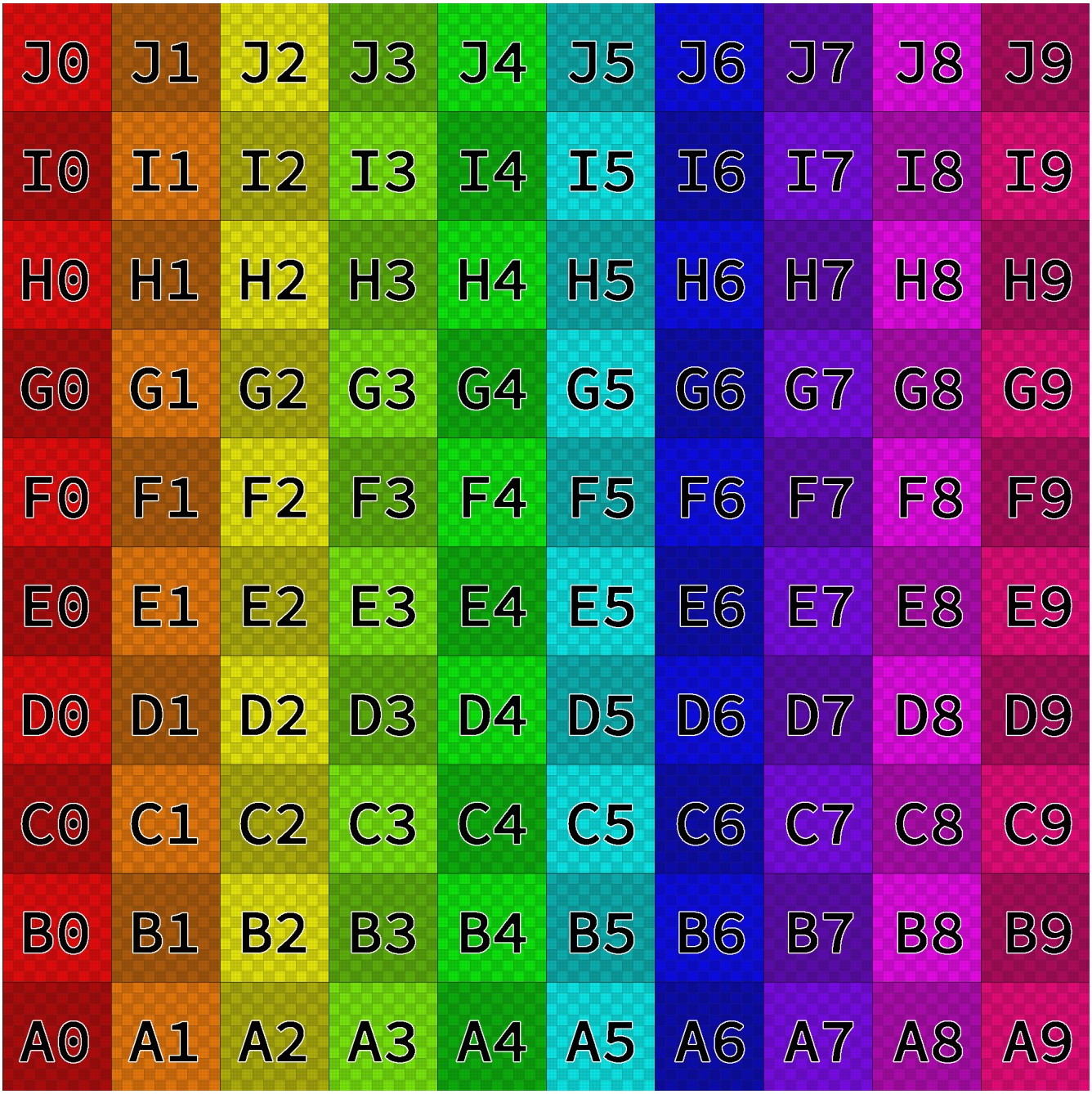
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9

D0 D1 D2 D3 D4 D5 D6 D7 D8 D9

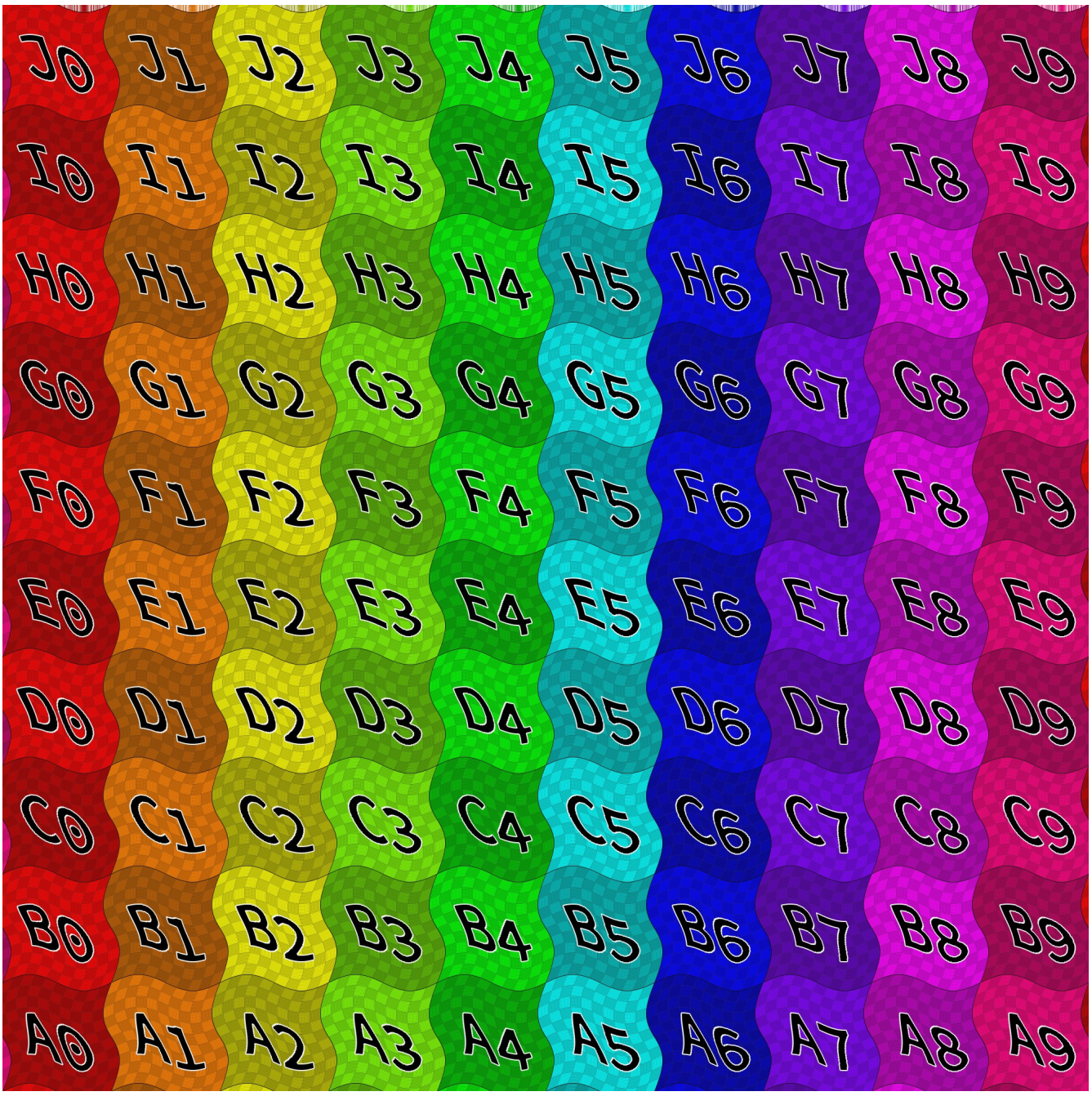
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9

B0 B1 B2 B3 B4 B5 B6 B7 B8 B9

A0 A1 A2 A3 A4 A5 A6 A7 A8 A9



Version to test tileability of a mesh:


J0 J1 J2 J3 J4 J5 J6 J7 J8 J9
I0 I1 I2 I3 I4 I5 I6 I7 I8 I9
H0 H1 H2 H3 H4 H5 H6 H7 H8 H9
G0 G1 G2 G3 G4 G5 G6 G7 G8 G9
F0 F1 F2 F3 F4 F5 F6 F7 F8 F9
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9
B0 B1 B2 B3 B4 B5 B6 B7 B8 B9
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9