|  |  |  |
| --- | --- | --- |
| Controller | pawn |  |
| Provides control  Interprets input  No collision | Has collision  Has model  Has pawn state |  |
|  |  |  |

PlayerControllerClass=class'UTGame.UTPlayerController'

controller

pawn

controller c

pawn p

p = c.pawn;

if (p!= none && p.health >=1)

Contorller.possess(pawn)

controller.unpossess()

set !bUseClassicHUD to true

if use own HUD type

playcontroller has myHUD

controller -> pawn: controller.pawn

pawn -> controller pawn.controller

contonroller -> hud: controller.hud

Exec functions

member func wqualfier exec

can be called from console

can bind input to it

keyword out

has lifetime more than function

multiple returns

udk make && L6-Map.udk –log

udkgames script

roll=y -> pitch=x -> yaw=z

UDK: 360 degree = 65536 = 2 power 16

Short 2 bytes

Range = -32768 to 32767

Positive axis pointing at u, positive rotator is clockwise

When cast vector into rotator, the roll of rotator is lost

R rotator(v);

November 3, 2015

## Interface

Declare a set of public functions, eg functions

Useful for

* Gurantee implementing class declared functions
* Interface tagging

**Myinterface.uc**

Interface myInterface;

Vector function GetVect();

Float function add(float f1, float f2);

**Myactor.uc**

Class myActor extends actor implements myinterface;

**Object**

Actors can be put in a map

Object is a non placeable unreal script class

W8object.uc

Class w8object extends object;

Var w8object v;

Function creator() {

Local w8Object;

v = new class’W80Object’;

}

Function remove() {

V = none;

}

Object outer;

Playerinput

Input bindings macros

Command can be :

Exec functions

Piped exec functions

Piped other macros

Modifiers + exec function or variables

**Class literal:**

Class’<package>.<classname>’

local Texture2D myTex;

myTex.SizeX;

in

x,y,z

out x, y coordinate depth z

(

<0 cord in front

0 coord is on plane of cam

>0 coord behind

)

if want to know cor visibility

z>0

x>0 , y>0

x<hud.sizex

y<hud.sizey

deproject

world pos

world direction

Actor/component

Creating placealbe actors

Declare class as placealbe

Declare catg using classgroup

Componets

Virtual object defineing interaction b/w engine resource and actor

Component relies on actors for:

Position

Rotation

Component list

Important component classes

StaticMeshComponent

skeletalMeshcomponent

var array<int> myflexInt;

var int myInt[10]

## Collision

Touch vs. Block

Block

Actors cannot pass through each other

|  |  |
| --- | --- |
| Touch | Block |
| Actors pass through each other, but interaction still note | Actors cannot pass through each other |
| Event: touch upon initial interaction  Untouch when no longer intersect | Event: bump |
|  |  |

Collison detection

Collision resolution

bCollideActors set to false for no collision

bBlockActors //whether block other actors

## Inventory

Every pawn has a set of inventory

Handled by inventory manager

Every pawn has an instance of inventory manager

Each pawn class can set its own inventory manager class

Inventory manager holds a list of inventory items

Not allowed to 2 items of same item type

Ensure each item is unique by type

simulated event Inventory FindInventoryType(class<Inventory> DesiredClass, optional bool bAllowSubclass)

if found class, return instance

else, return none

simulated function Inventory CreateInventory(class<Inventory> NewInventoryItemClass, optional bool bDoNotActivate)

on success, creates and retuns item

failure: retuns none

InvManager

Instance of this pawn’s inv manager

## Weapons

Weapon <- basic framework for logic that activates on firing

Utweapon <- more functionality, for animation

Weapon from inventory

Weapons have firemodes (0 or more)

Each firemode’s behavior can be categorized as follows

Shoots projectile

Performs hitscan (tracing)

Custom

Weapon has ammo count

Projectile

UTProjectile

Location <- vector

Rotation <- rotator

**Velocity <- vector**

**Acceleration <- vector**

Utprojectile

**Component**

Link engine resource with actor

**Weapon basic properties**

WeaponFire Types

Weapon projectile array

Fire interval

Shot cost

Ammo count

Max ammo count

Mesh

## Projectile

FireAmmunition()

Consume Ammo

CurrenFireMode

GetPhysicalFireStartLoc

GetAdjustedAim

## AI

To create AI

* Build sane path nodes
* Build spawner for ai
* Build ai controller

## Instigator

// Pawn responsible for damage caused by this actor.

http://udn.epicgames.com/Three/GametypeTechnicalGuide.html#GameInfo Class

## Hitscan

Hitlocation

Hit normal

Other (the thing they hit)

Local pawn p;

P = …;

For each p.traceactors(class’pawn’);

{

}

Implement on weapon

Override instantfirestart trace

Instant fire end trace

Processinstanthit

## Review

Iterator

Distance

Vector rotation

p = G6PlayerController (PlayerOwner);

p\_input = G6PlayerInput (p.PlayerInput);

out\_CamRot = rotator(normal(p.Pawn.Location - out\_CamLoc));

foreach AllActors( class'G6Spawner', S )

distance = VSize(B.Location - Pawn.location);

bCollideActors

bBlockActors