Niculescu Tiberiu-Gabriel

Pac-Man

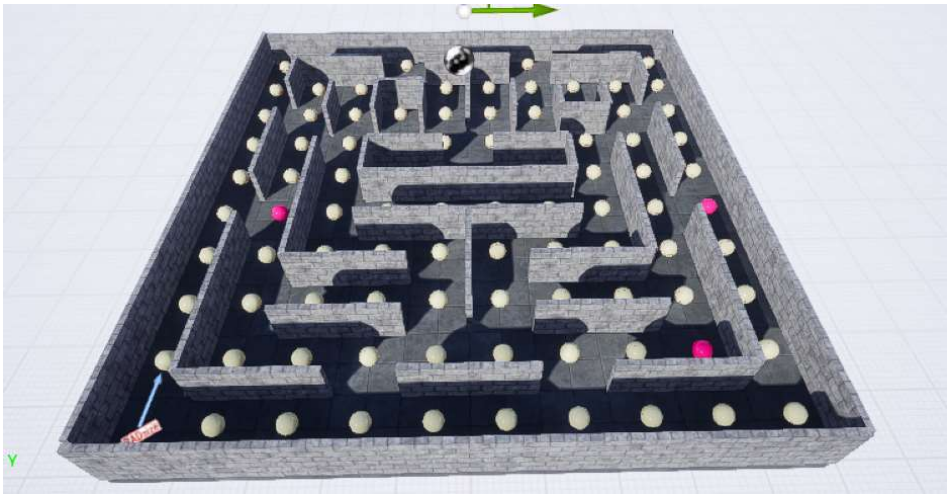
Pac-Man este un joc aparut in anul 1980, produs de compania Namco.

Este un joc cu un principiu simplu: Jucatorul preia controlul caracterului nostru Pac-Man printr-un labirint, cu scopul de a consuma toate colectabilele de pe harta. Impedimentul il reprezinta cei 4 inamici care in cazul coliziunii cu Pac-Man, ii vor scadea din viata caracterului. Dupa pierderea a 3 vieti, jocul se termina.

Am ales crearea acestui joc in Unreal Engine, in C++, in ideea de a il face sa arate mai bine si de a ma familiariza cu aplicatia in care doresc sa mai lucrez pe viitor.

Componentele de baza ce trebuiesc proiectate in cadrul acestui proiect:

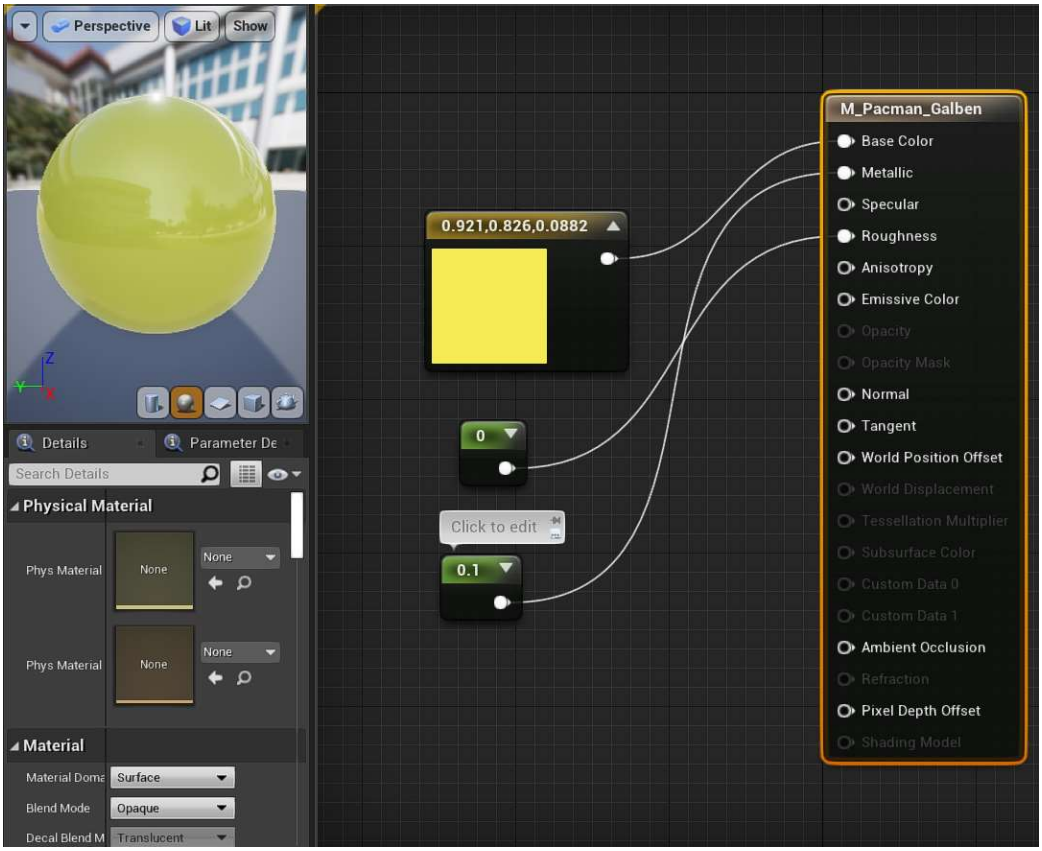
Harta:

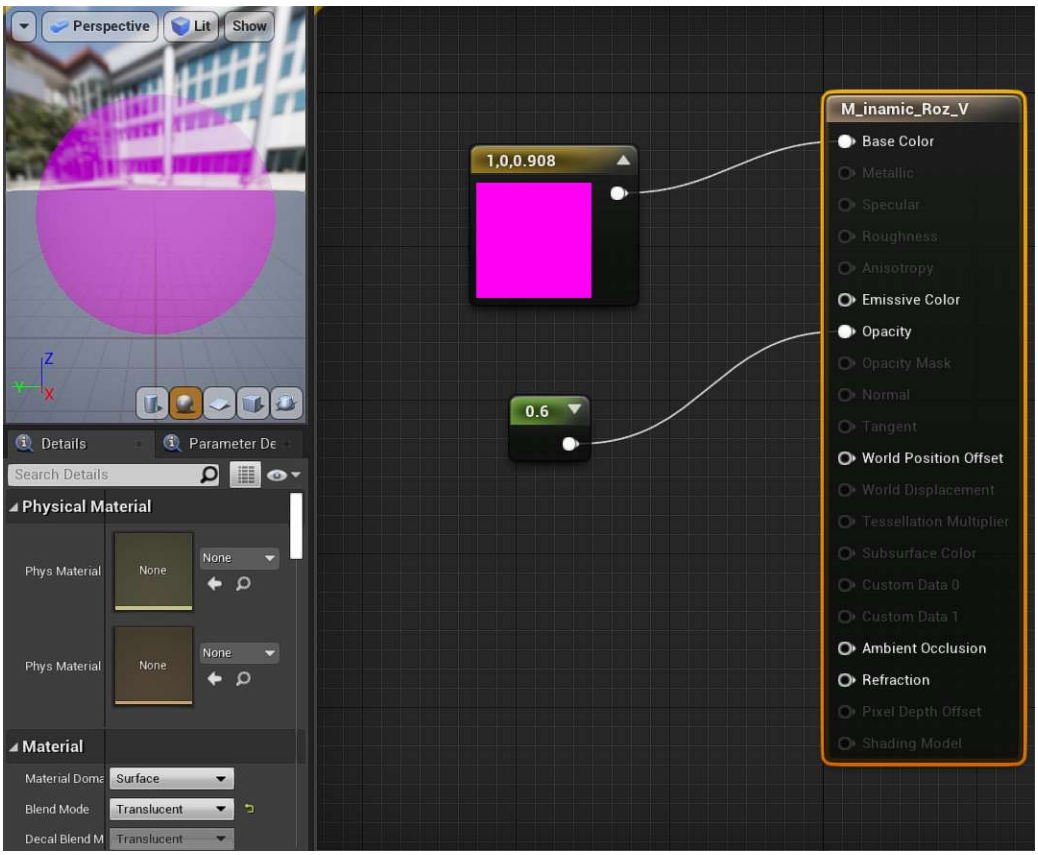


Aceasta a fost construita folosind Brush-uri cu forme cubice, pozitionate si modificate astfel incat caracterele noastre sa nu poata sa treaca prin/peste ele, sau sa cada prin harta. Materialele aplicate dupa preferinta.

Materiale:

Avem nevoie de materiale pentru caracter, colectabile, inamici si pentru cazul in care una din colectabilele consumate este speciala, inamicii devenind astfel transparenti si vulnerabili.


Perspective
Lit
Show
Details
Parameter De
Search Details
Physical Material
Phys Material
None
None
Phys Material
None
None
Material
Material Doma
Surface
Blend Mode
Opaque
Decal Blend M
Translucent
0.921,0.826,0.0882
0
Click to edit
0.1
M_Pacman_Galben
Base Color
Metallic
Specular
Roughness
Anisotropy
Emissive Color
Opacity
Opacity Mask
Normal
Tangent
World Position Offset
World Displacement
Tessellation Multiplier
Subsurface Color
Custom Data 0
Custom Data 1
Ambient Occlusion
Refraction
Pixel Depth Offset
Shading Model


Perspective
Lit
Show
Details
Parameter De
Search Details
Physical Material
Phys Material
None
None
Phys Material
None
None
Material
Material Doma
Surface
Blend Mode
Translucent
Decal Blend M
Translucent
1,0,0.908
0.6
M_inamic_Roz_V
Base Color
Metallic
Specular
Roughness
Anisotropy
Emissive Color
Opacity
Opacity Mask
Normal
Tangent
World Position Offset
World Displacement
Tessellation Multiplier
Subsurface Color
Custom Data 0
Custom Data 1
Ambient Occlusion
Refraction
Pixel Depth Offset
Shading Model

Clasele:

1) Colectabilele

Clasa de colectabile este prima clasa pe care o implementam. Sunt obiecte ce mostenesc functionalitati de la clasa Actor, care la coliziunea cu jucatorul sunt distruse. De asemenea, 4 dintre acestea trebuie sa fie SuperColectabile, ducand la vulnerabilitatea inamicilor.

Collectable.cpp:

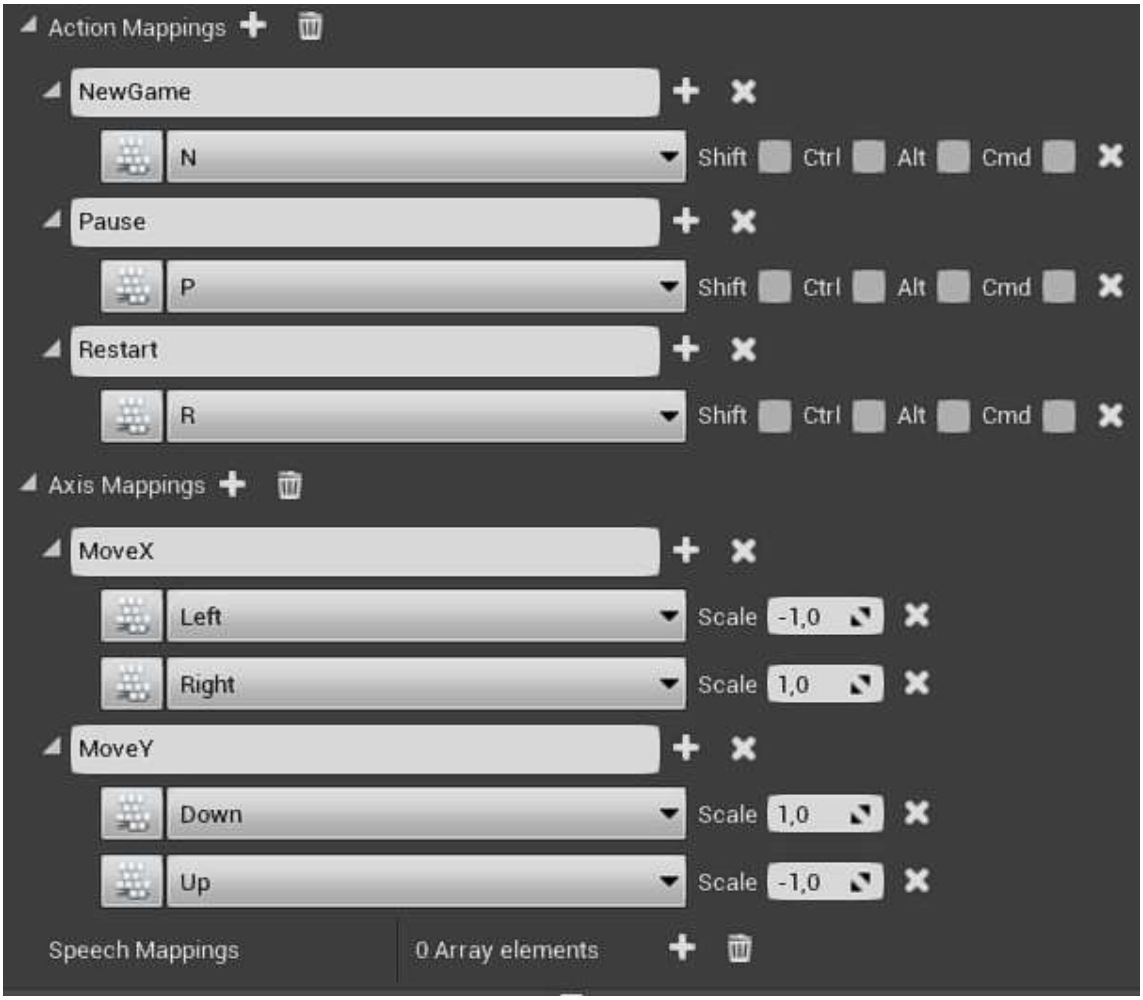
```
#include "Collectable.h"
#include "D:\Director lucru\SMPProiect\PacMan\Source\PacMan\PacMan.h"


 // Sets default values
ACollectable::ACollectable()
{
    // Set this actor to call Tick() every frame.  You can turn this off to improve performance if you don't need it.
    PrimaryActorTick.bCanEverTick = false;
    SetActorEnableCollision(true);
    BaseCollisionComponent = CreateDefaultSubobject<USphereComponent>(TEXT("BaseCollisionComponent"));
    CollectableMesh = CreateDefaultSubobject<UStaticMeshComponent>(TEXT("CollectableMesh"));
    CollectableMesh->AttachTo(BaseCollisionComponent);
    static ConstructorHelpers::FObjectFinder<UStaticMesh> Sphere(TEXT("StaticMesh'/Engine/BasicShapes/Sphere.Sphere'"));
    CollectableMesh->SetStaticMesh(Sphere.Object);
    CollectableMesh->SetWorldScale3D(FVector(0.3, 0.3, 0.3));
    BaseCollisionComponent->SetSphereRadius(14);
}

void ACollectable::BeginPlay()
{
    Super::BeginPlay();

    BaseCollisionComponent->SetCollisionProfileName(TEXT("Collectable"));
    CollectableMesh->SetCollisionProfileName(TEXT("NoCollision"));
}
```

Collectable.h:

```
#include "CoreMinimal.h"
#include "GameFramework/Actor.h"
#include <Runtime\Engine\Classes\Components\SphereComponent.h>
#include "Collectable.generated.h"


UCLASS()
class PACMAN_API ACollectable : public AActor
{
    GENERATED_BODY()
        //Line skip 1
        //Line skip 2
public:
    // Sets default values for this actor's properties
    ACollectable();

    virtual void BeginPlay() override;

    UPROPERTY(EditDefaultsOnly, Category = Collectable)
        USphereComponent* BaseCollisionComponent;

    UPROPERTY(EditDefaultsOnly, Category = Collectable)
        UStaticMeshComponent* CollectableMesh;

    UPROPERTY(EditAnywhere, Category = Collectable)
        bool bIsSuperCollectable;
```

Caracterul:

Dupa crearea clasei de colectabile si pozitionarea acestora ca in imaginea de mai sus (Harta), la aceeasi inaltime si la distante egale intre ele, implementam caracterul nostru Pac-Man. Acesta mosteneste de la clasa Character, primul lucru pe care il implementam fiind miscarea acestuia. Din fereastra de Input in UnrealEngine, setam tastele pentru controlul jucatorului (si cele folosite pentru actiuni daca tot suntem aici):


Action Mappings
NewGame
N
Shift
Ctrl
Alt
Cmd
Pause
P
Shift
Ctrl
Alt
Cmd
Restart
R
Shift
Ctrl
Alt
Cmd
Axis Mappings
MoveX
Left
Scale
-1,0
Right
Scale
1,0
MoveY
Down
Scale
1,0
Up
Scale
-1,0
Speech Mappings
0 Array elements

```cpp
void APacManCharacter::SetupPlayerInputComponent(UInputComponent* PlayerInputComponent)
{
    Super::SetupPlayerInputComponent(PlayerInputComponent);
    // bind the input from the player
    InputComponent->BindAxis("MoveX", this, &APacManCharacter::MoveXAxis);
    InputComponent->BindAxis("MoveY", this, &APacManCharacter::MoveYAxis);
    InputComponent->BindAction("NewGame", IE_Pressed, this, &APacManCharacter::NewGame);
    InputComponent->BindAction("Pause", IE_Pressed, this, &APacManCharacter::Pause);
    InputComponent->BindAction("Restart", IE_Pressed, this, &APacManCharacter::RestartGame);

}

void APacManCharacter::MoveXAxis(float AxisValue)
{
    CurrentVelocity.X = AxisValue;
    AddMovementInput(CurrentVelocity);

}

void APacManCharacter::MoveYAxis(float AxisValue)
{
    CurrentVelocity.Y = AxisValue;
    AddMovementInput(CurrentVelocity);
}
```

```cpp
void APacManCharacter::NewGame()
{
    if (GameMode->GetCurrentState() == EGameState::EMenu) {
        GameMode->SetCurrentState(EGameState::EPlaying);
    }
}

void APacManCharacter::Pause()
{
    if (GameMode->GetCurrentState() == EGameState::EPlaying) {
        GameMode->SetCurrentState(EGameState::EPause);
    }
    else if (GameMode->GetCurrentState() == EGameState::EPause) {
        GameMode->SetCurrentState(EGameState::EPlaying);
    }
}

void APacManCharacter::RestartGame()
{
    GetWorld()->GetFirstPlayerController()->ConsoleCommand(TEXT("RestartLevel"));
}
```

Folosim pentru acestea din urma o clasa de Enums definita in fisierul nostru PacManGameModeBase, unde ne asiguram ca ceea ce implementam se aplica

jocului nostru si unde ne ocupam de comportamentul programului in functie de starea in care se afla.

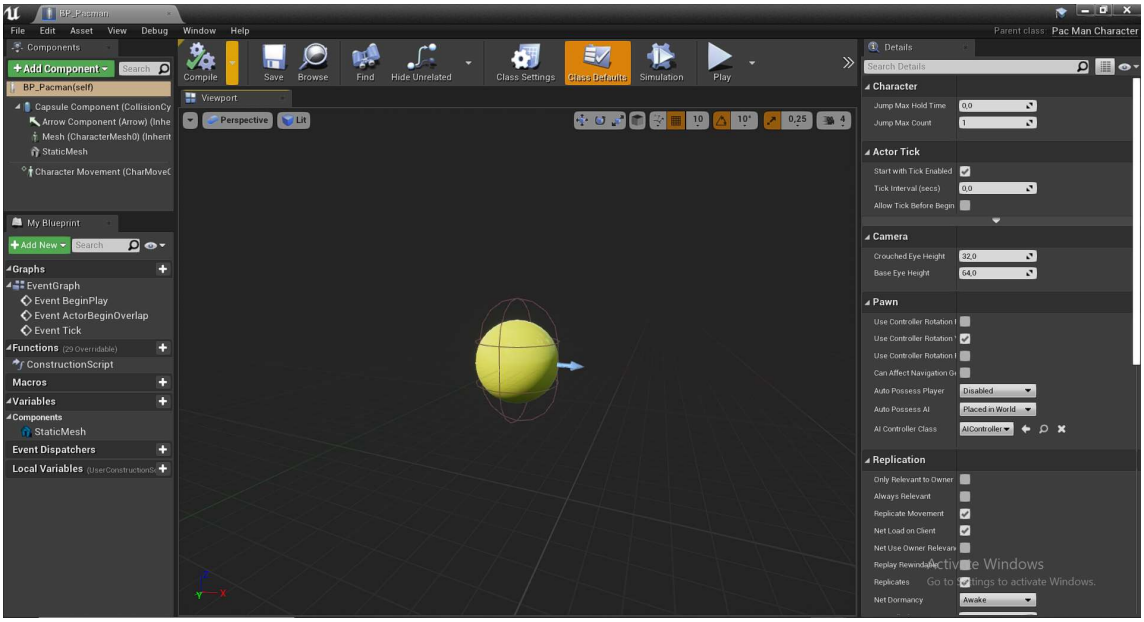
```
enum class EGameState : short
{
    EMenu,
    EPlaying,
    EPause,
    EWin,
    EGameOver
};
```

O alta functie importanta a caracterului nostru este aceea de a distruge colectabilele in momentul coliziunii, pentru care am folosit o functie delegat, si de asemenea sa tinem cont de cate colectabile mai are nevoie jucatorul pentru a castiga.

```
void APacManCharacter::OnCollision(class UPrimitiveComponent* OverlappedComp, class AActor* OtherActor, UPrimitiveComponent* OtherComp, int32 Ot
{

    if (GameMode->GetCurrentState() == EGameState::EPlaying)
    {

        if (OtherActor->IsA(ACollectable::StaticClass()))
        {
            if (--CollectablesToEat == 0) {
                GameMode->SetCurrentState(EGameState::EWin);
            }

            OtherActor->Destroy();
        }
    }
}
```

```
void APacManCharacter::BeginPlay()
{
    Super::BeginPlay();
    GameMode = Cast<APacManGameMode>(UGameplayStatics::GetGameMode(this));
    GetCapsuleComponent()->SetCollisionProfileName(TEXT("Player"));
    GetCapsuleComponent()->OnComponentBeginOverlap.AddDynamic(this, &APacManCharacter::OnCollision);

    for (TActorIterator<ACollectable> CollectableItr(GetWorld()); CollectableItr; ++CollectableItr)
    {
        CollectablesToEat++;
    }
    StartPoint = GetActorLocation();
    Lives = 3;

}
```

Caracerul nostru este controlat de jucator printr-un model Blueprint ce are ca parinte clasa PacManCharacter:




```
APacManGameMode::APacManGameMode()
{

    static ConstructorHelpers::FClassFinder<APawn> PlayerPawnOb(TEXT("'/Game/BP_Pacman'"));
    if (PlayerPawnOb.Class != NULL)
    {
        DefaultPawnClass = PlayerPawnOb.Class;
    }
    HUDClass = APacManHud::StaticClass();
}
```

Inamicii:

Pentru inamici vom folosi doua clase:

- o clasa de tip Character ce vor fi inamicii nostri propriu-zisi, unde le vom seta forma, materialul, marimea, vulnerabilitatea si coliziunile;

- o clasa de tip AI ce va coordona miscarile inamicilor, ce vor fi aleatoare.

Am ales forma de cilindru pentru inamicii nostri.

Pentru miscarea aleatoare a inamicilor ne folosim de sistemul de navigatie Unreal Engine, pentru care vom amplasa pe harta noastra o componenta de ti NavMeshBoundsVolume, ce va delimita pe unde se pot misca caracterele noastre. Apasand tasta P aceasta se poate observa prin culoarea verde.

Clasa HUD:

Aici am implementat interfata cu utilizatorul. Ne va arata in timpul jocului numarul de vieti pe care le mai avem si starea jocului in care ne aflam. Este inclusa in GameMode-ul nostru pentru a functiona, dupa cum se poate vedea mai sus.