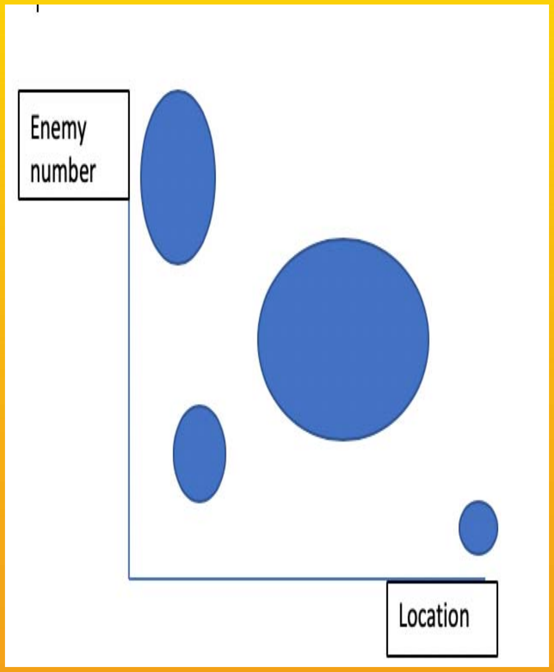
Enemy Hunt: Cluster Analysis for Decision Making

Yordanos Bezabeh
COSC 729

Introduction

- Decision making is an essential part of human life
- It is crucial to know the decision-making techniques that enables us to make the right choices
- Cluster analysis is a method of grouping data into meaningful groups to make conscious decision

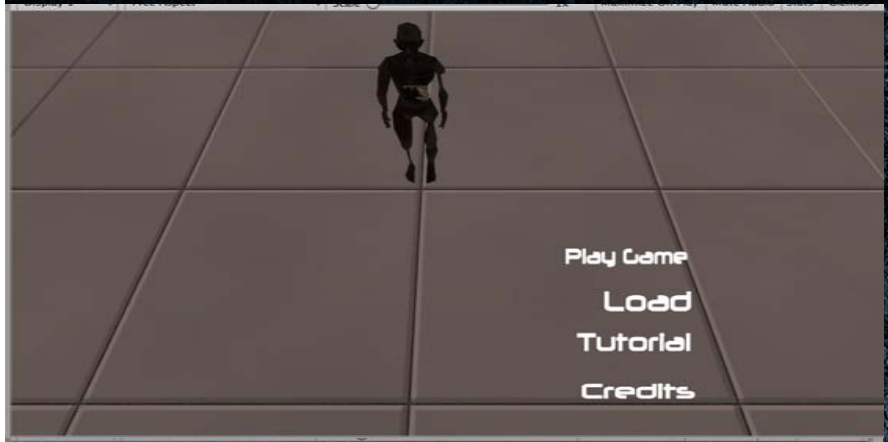


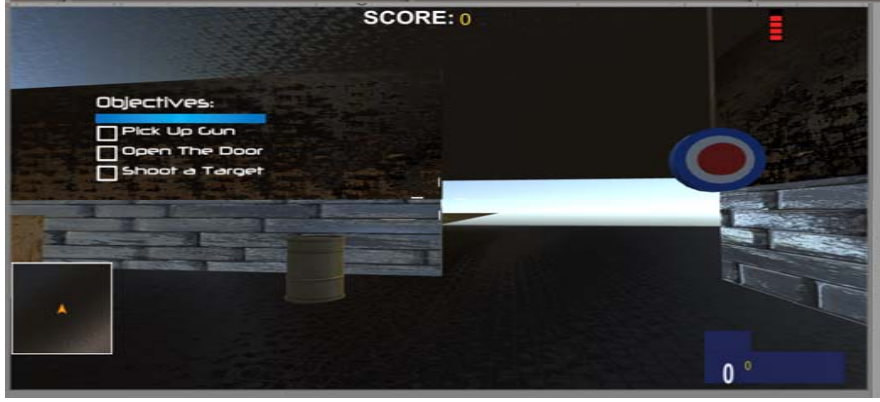
Goals and Objectives

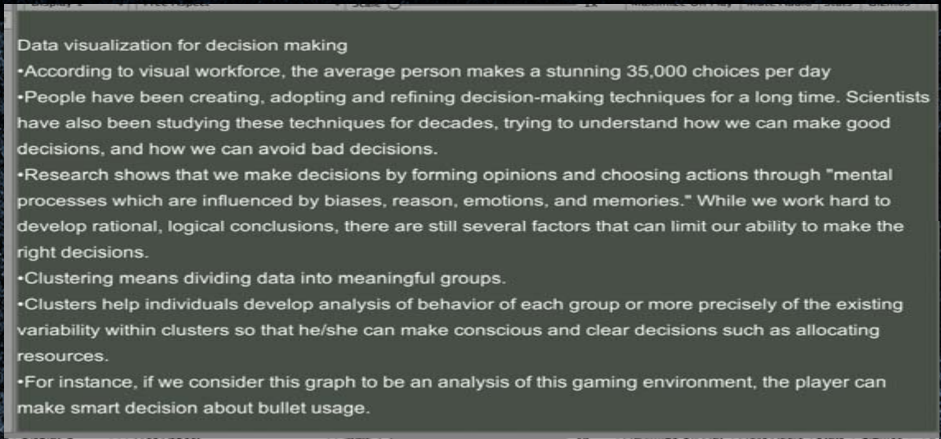
- Demonstrate the use of Cluster Analysis in making decisions
- Increase the player’s decision-making ability
- The goal is to navigate through the environment and try to kill as much enemy as possible before running out of ammo
- Its about taking the time to accomplish the goal

Environment and Functions

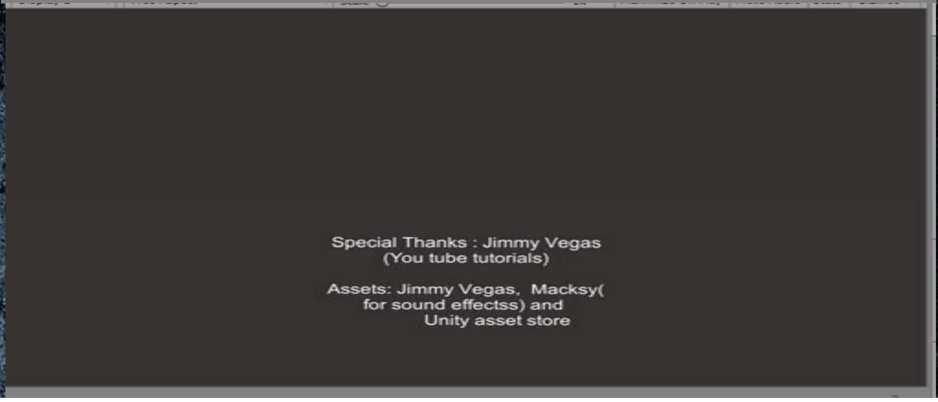
- Has four different scenes: Play Game, Load, Tutorial and Credit
- Play game and Load takes the player to the main game environment
- Tutorial takes the player to a page that contains information about Cluster Analysis and its application in this game
- Credit : It simply recognizes people who has directly or contributed to this project






Data visualization for decision making

- According to visual workforce, the average person makes a stunning 35,000 choices per day
- People have been creating, adopting and refining decision-making techniques for a long time. Scientists have also been studying these techniques for decades, trying to understand how we can make good decisions, and how we can avoid bad decisions.
- Research shows that we make decisions by forming opinions and choosing actions through "mental processes which are influenced by biases, reason, emotions, and memories." While we work hard to develop rational, logical conclusions, there are still several factors that can limit our ability to make the right decisions.
- Clustering means dividing data into meaningful groups.
- Clusters help individuals develop analysis of behavior of each group or more precisely of the existing variability within clusters so that he/she can make conscious and clear decisions such as allocating resources.
- For instance, if we consider this graph to be an analysis of this gaming environment, the player can make smart decision about bullet usage.


Special Thanks : Jimmy Vegas
(You tube tutorials)

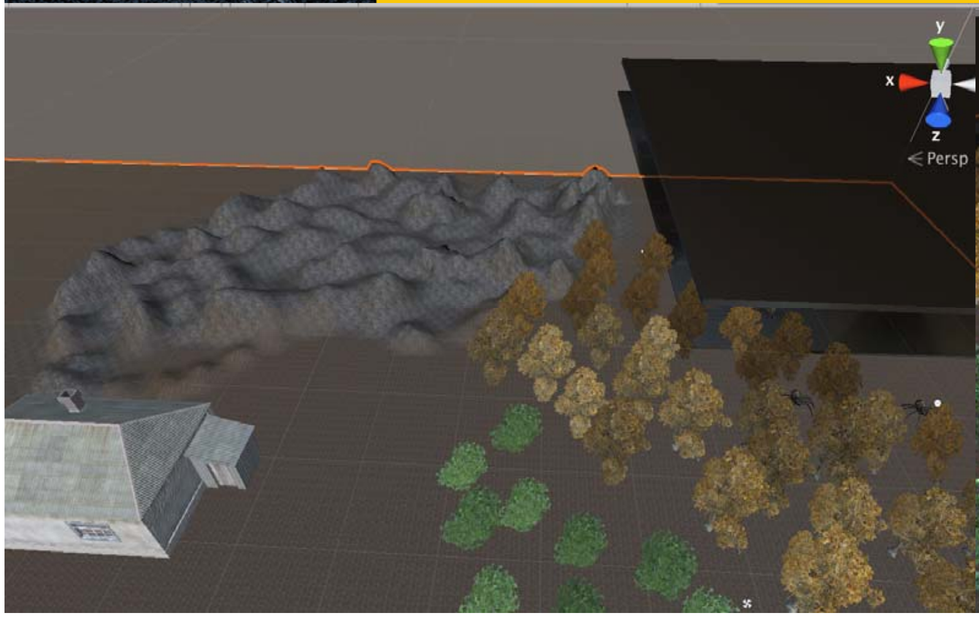
Assets: Jimmy Vegas, Macksy(
for sound effectss) and
Unity asset store

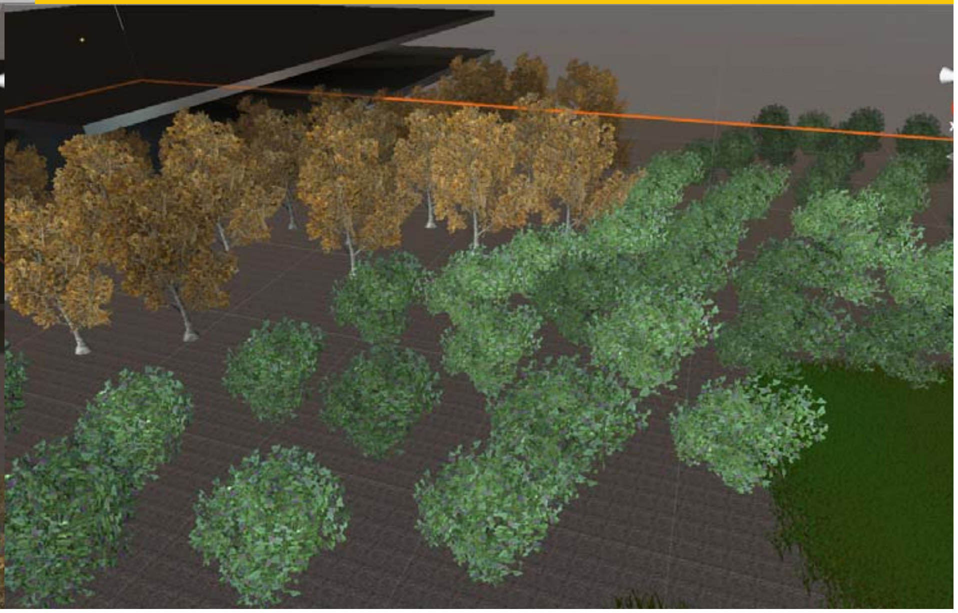
Environment

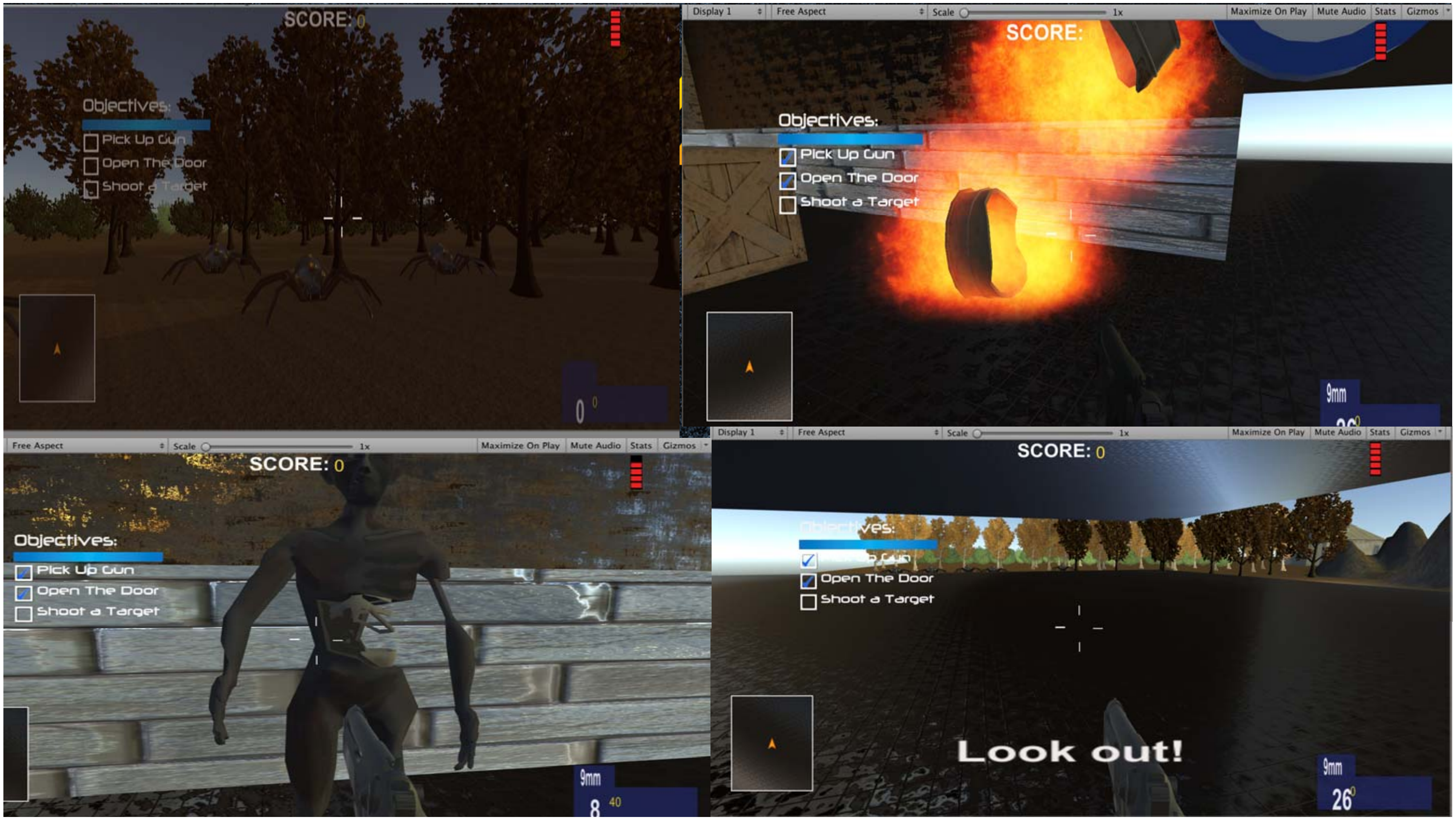
- The environment has two settings: Indoor and outdoor
- Indoor: A bunker like setup

Environment

- Outdoors






SCORE: 0
Objectives:
Pick Up Gun
Open The Door
Shoot a Target
Display 1
Free Aspect
Scale
1x
Maximize On Play
Mute Audio
Stats
Gizmos
9mm
8 40
26 0
Look out!

Software Requirement

- The development software includes:
  - Unity 2018
  - Visual Studio
- Software Required:
  - macOS 10.12. 6

Modeling

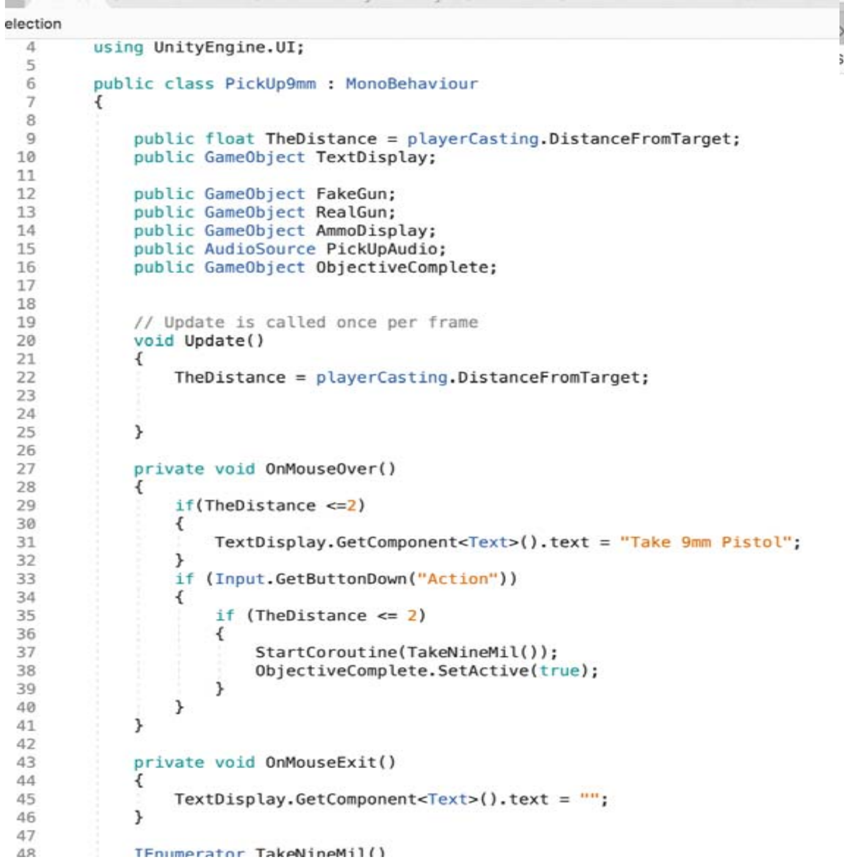
- **Vision**: Use 3d models and textures to give more realistic environmental feel
  - Arrangement of enemy models to best portray clustering Vs. distribution
  - Incorporate indoor and outdoor for a more realistic feeling
- **Sound**: sound effects and background music
  - Enemy attack, explosion, warning

Modeling

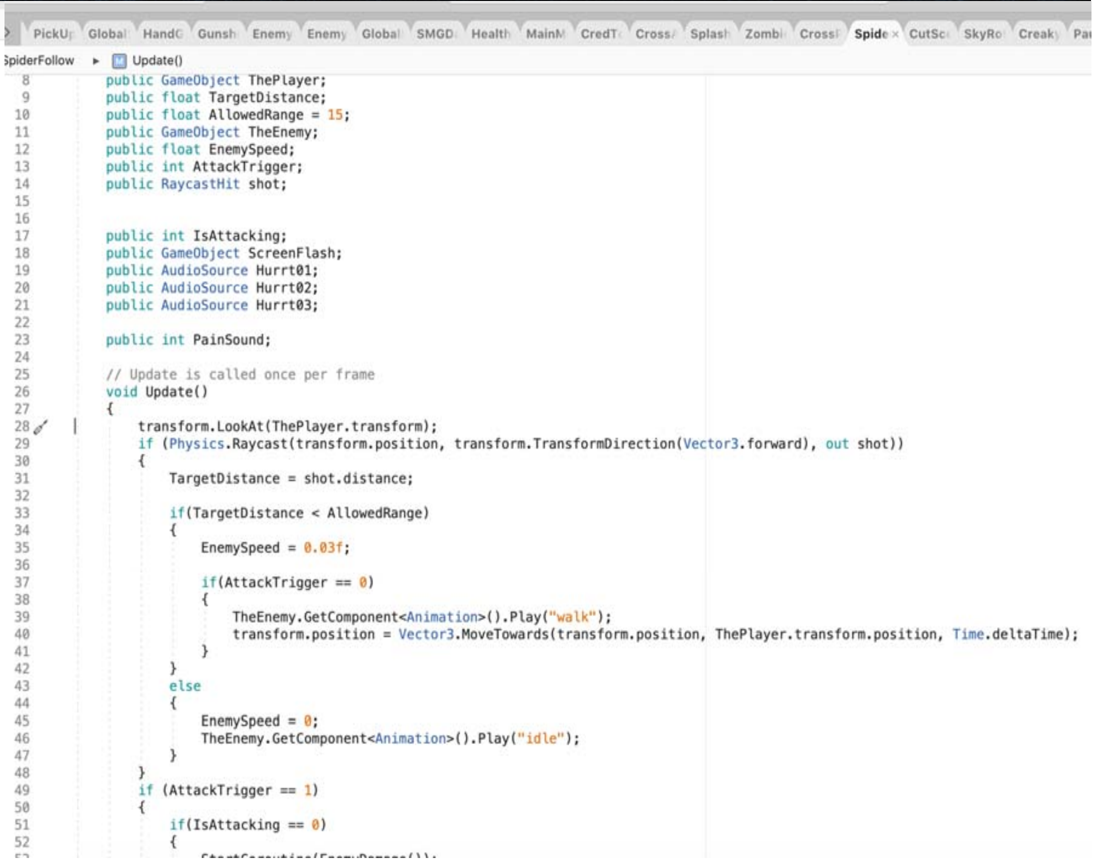
- **Sensors**: Proximity sensors for enemy, opening door, and picking up guns
- **Interactivity**: Picking up gun, picking up ammos, and killing enemies
- **Animations**: there are multiple animations
  - Enemy: attack, idle and die
  - Player: Idle, jump and walk
  - Door: opening animation

Modeling

- **Coding**: multiple C# codes have been applied. These codes include
  - a code for handgun damage
  - A code for enemy attack and follow
  - A code to control the player health
  - A code to add score
  - A code to go from one scene to another automatically



```
using UnityEngine.UI;

public class PickUp9mm : MonoBehaviour
{

    public float TheDistance = playerCasting.DistanceFromTarget;
    public GameObject TextDisplay;

    public GameObject FakeGun;
    public GameObject RealGun;
    public GameObject AmmoDisplay;
    public AudioSource PickUpAudio;
    public GameObject ObjectiveComplete;


    // Update is called once per frame
    void Update()
    {
        TheDistance = playerCasting.DistanceFromTarget;


    }

    private void OnMouseOver()
    {
        if(TheDistance <=2)
        {
            TextDisplay.GetComponent<Text>().text = "Take 9mm Pistol";
        }
        if (Input.GetButtonDown("Action"))
        {
            if (TheDistance <= 2)
            {
                StartCoroutine(TakeNineMil());
                ObjectiveComplete.SetActive(true);
            }
        }
    }

    private void OnMouseExit()
    {
        TextDisplay.GetComponent<Text>().text = "";
    }

    IEnumerator TakeNineMil()
```



PickU Global HandG Gunsh Enemy Enemy Global SMGD Health MainM CredT Cross Splash Zombi Cross Spide CutSc SkyRo Creak Pa
SpiderFollow ▸ Update()

```
    public GameObject ThePlayer;
    public float TargetDistance;
    public float AllowedRange = 15;
    public GameObject TheEnemy;
    public float EnemySpeed;
    public int AttackTrigger;
    public RaycastHit shot;


    public int IsAttacking;
    public GameObject ScreenFlash;
    public AudioSource Hurrt01;
    public AudioSource Hurrt02;
    public AudioSource Hurrt03;

    public int PainSound;

    // Update is called once per frame
    void Update()
    {
        transform.LookAt(ThePlayer.transform);
        if (Physics.Raycast(transform.position, transform.TransformDirection(Vector3.forward), out shot))
        {
            TargetDistance = shot.distance;

            if(TargetDistance < AllowedRange)
            {
                EnemySpeed = 0.03f;

                if(AttackTrigger == 0)
                {
                    TheEnemy.GetComponent<Animation>().Play("walk");
                    transform.position = Vector3.MoveTowards(transform.position, ThePlayer.transform.position, Time.deltaTime);
                }
            }
            else
            {
                EnemySpeed = 0;
                TheEnemy.GetComponent<Animation>().Play("idle");
            }
        }
        if (AttackTrigger == 1)
        {
            if(IsAttacking == 0)
            {
```

Thank you