ICAT
Design & Media College

BONAFIDE CERTIFICATE

This is to certify that record of course work is a bonafide work done by **Ajil Pappachan**, ID No.: **2018UG03077**, in partial fulfillment of the requirements for the **2nd year B.Sc. Game Programming** during the academic year **2019 – 2020** is the original work of the candidate.

Submitted for the **ARTIFICIAL INTELLIGENCE** assessment held on _______________.

_________________ _________________

Verified By **Staff In-Charge**

GAME DESIGN DOCUMENT

Hide n' Seek

INDEX

INTRODUCTION

Hide n Seek is a 3d casual game for the Windows platform. The game is based on the traditional children's game "Hide and go Seek", recreated to implement Artificial Intelligence as Non-Player Characters (NPCs). The player can move around using WASD keys and has to locate other players and come back to the flag post to tag them. The player wins if he can tag every NPC and loses if an NPC reaches the flagpost first.

GAME DESIGN

Game Name	:	Hide n' Seek
Genre	:	3D Casual Game
Target Audience	:	Teenagers, young adults
Target Platform	:	Microsoft Windows

SOFTWARE SPECIFICATIONS

Operating System: Microsoft Windows

API: OpenGL

HARDWARE SPECIFICATIONS

Intel i3 Processor or equivalent

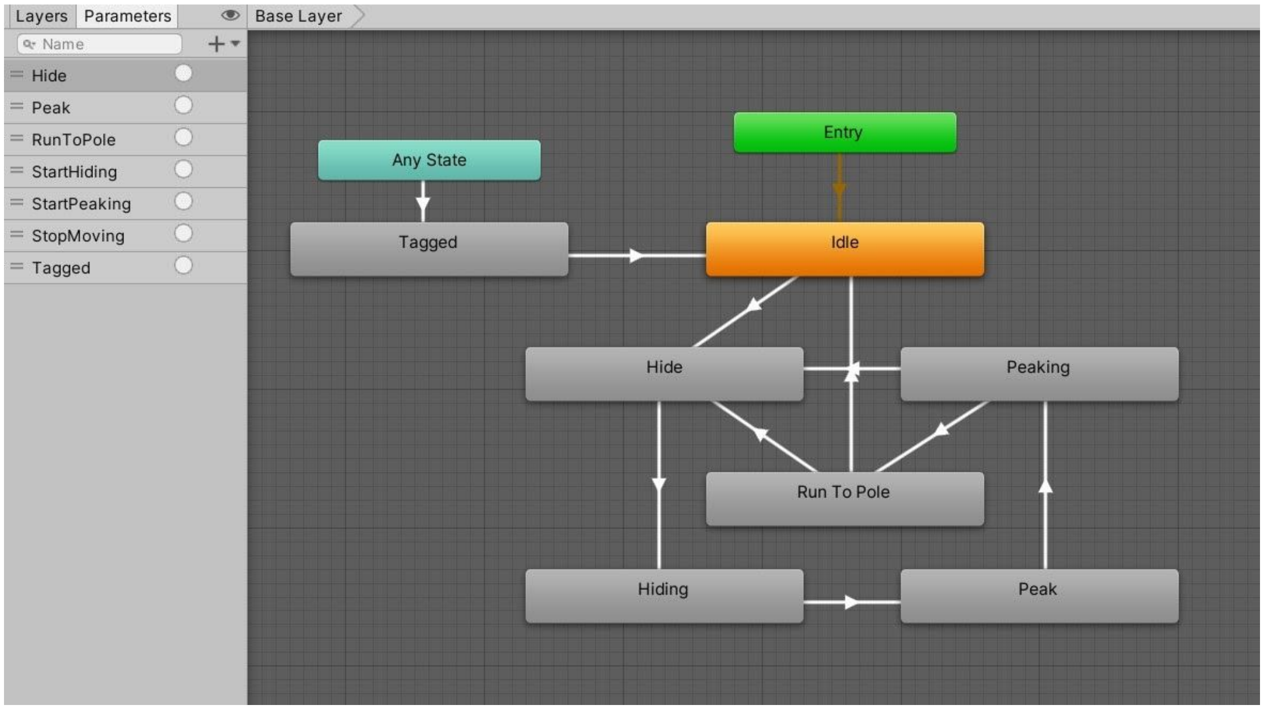
5 GB Free Space

4 GB RAM

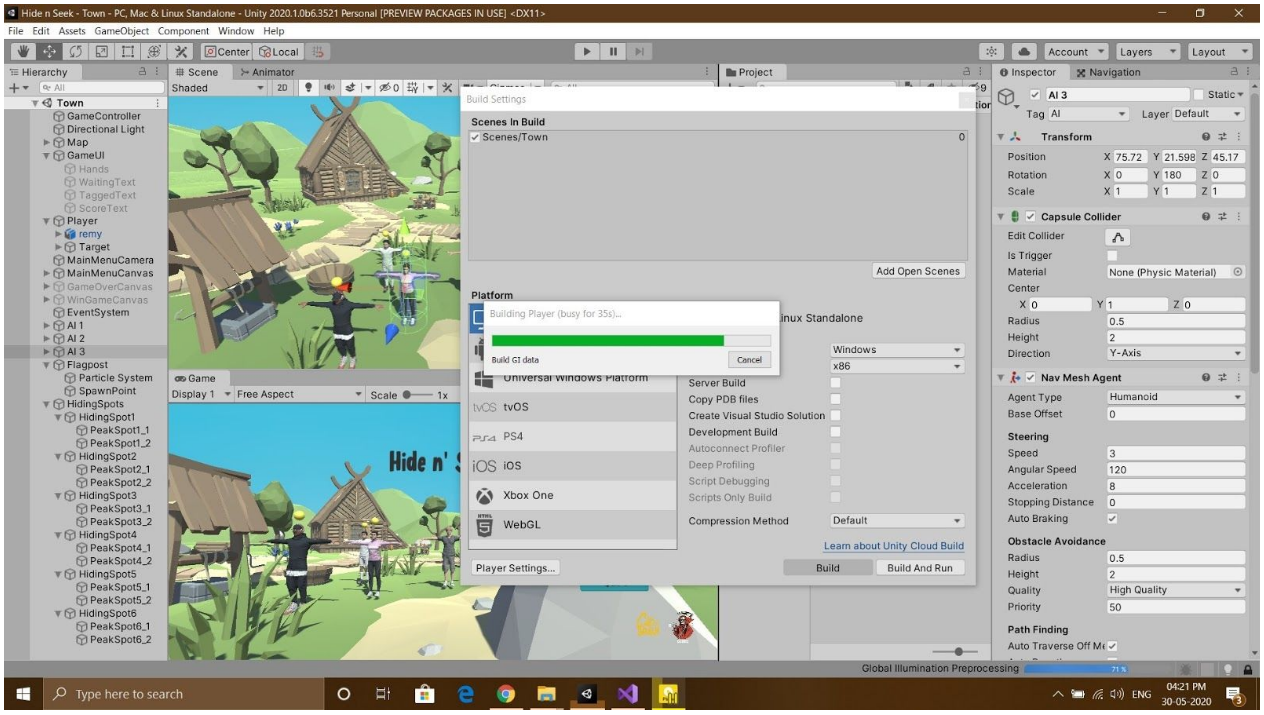
BUILT ENVIRONMENT

Unity Game Engine
Mixamo (For Characters)
Unity Marketplace (For Level Map)
Photoshop (For Sprites)
FreeSFX (For Music and Sound)

DEVELOPMENT SNAPSHOTS



AI Behavior Tree



Scenes/Town	0

Target Platform	Windows
Architecture	x86
Server Build	
Copy PDB files	
Create Visual Studio Solution	
Development Build	
Autoconnect Profiler	
Deep Profiling	
Script Debugging	
Scripts Only Build	
Compression Method	Default

AI 3 Static
Tag AI Layer Default

Transform

	X	Y	Z
Position	75.72	21.598	45.17
Rotation	0	180	0
Scale	1	1	1

Capsule Collider

Edit Collider	
Is Trigger	
Material	None (Physic Material)
Center	X 0 Y 1 Z 0
Radius	0.5
Height	2
Direction	Y-Axis

Nav Mesh Agent

Agent Type	Humanoid
Base Offset	0
Steering	
Speed	3
Angular Speed	120
Acceleration	8
Stopping Distance	0
Auto Braking	✓
Obstacle Avoidance	
Radius	0.5
Height	2
Quality	High Quality
Priority	50
Path Finding	
Auto Traverse Off Me	✓

Building the Final Game

Code Snippets

The following is the script for the game controller which controls the main functionalities of the game and sends information to other GameObjects and resources:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class GameControllerScript : MonoBehaviour
{
    public Camera mainMenuCamera, playerCamera;

    public Canvas mainMenuCanvas;
    public Canvas gameOverCanvas;
    public Canvas winGameCanvas;

    public Text taggedText;
    public Text scoreText;
    int tagged;
    int score;
    public GameObject Hands;
    public Text waitText;
    float timeToWait;
    bool isWaiting;

    public GameObject player;
    public GameObject[] AIs;
    public GameObject[] HidingSpots;
    public GameObject spawnPoint;

    public bool isPlaying;
    public AudioSource soundEffects;
    public AudioClip PlayClip;
    public AudioClip WinClip;
    public AudioClip LoseClip;

    // Start is called before the first frame update
    void Start()
    {
        timeToWait = 10.0f;
        tagged = 0;
        score = 0;
        isWaiting = false;
        isPlaying = false;
        AIs = GameObject.FindGameObjectsWithTag("AI");
        HidingSpots = GameObject.FindGameObjectsWithTag("HidingSpot");
    }

    // Update is called once per frame
    void Update()
    {
        if (isPlaying && !isWaiting)
        {

player.GetComponent<PlayerControllerScript>().playerMovement();
            player.GetComponent<PlayerControllerScript>().checkPlayers();
        }
        if (isWaiting)
        {
            waiting();
        }
    }
    public void play()
    {
        playerCamera.gameObject.SetActive(true);
        mainMenuCamera.gameObject.SetActive(false);
        mainMenuCanvas.gameObject.SetActive(false);
        isPlaying = true;
        startWaiting();
        for (int i = 0; i < AIs.Length; i++)
        {
            AIs[i].GetComponent<AIScript>().hide(false);
        }

        soundEffects.clip = PlayClip;
        soundEffects.Play();

    }

    void startWaiting()
    {

        waitText.gameObject.SetActive(true);
        Hands.SetActive(true);
        isWaiting = true;
    }

    void waiting()
    {
        waitText.text = "Time To Wait : " + (int)timeToWait + " s";
        timeToWait -= Time.deltaTime;

        Debug.Log("WaitStart");
        if (timeToWait <= 0.0f)
        {
            isWaiting = false;
            waitText.gameObject.SetActive(false);
            Hands.SetActive(false);
            player.GetComponent<PlayerControllerScript>().canTag = true;
            scoreText.gameObject.SetActive(true);
            taggedText.gameObject.SetActive(true);
        }
    }
    public void updateTagged()
    {
        tagged++;
        taggedText.text = "Tagged : " + tagged;
    }

    public void updateScore()
    {
        score += tagged;
        tagged = 0;
        scoreText.text = "Score : " + score;
        taggedText.text = "Tagged : " + tagged;
        if(score == AIs.Length)
        {
            winGame();
        }
    }

    public void quit()
    {
        Application.Quit();
    }

    public void restart()
    {
        timeToWait = 10.0f;
        tagged = 0;
        score = 0;
        isWaiting = false;
        isPlaying = false;
        // SceneManager.LoadScene(0, LoadSceneMode.Single);
        SceneManager.LoadSceneAsync(1);
    }

    public void gameOver()
    {
        Time.timeScale = 0.0f;
        gameOverCanvas.gameObject.SetActive(true);

        Cursor.visible = true;
        Cursor.lockState = CursorLockMode.None;
        soundEffects.clip = LoseClip;
        soundEffects.Play();
    }

    public void winGame()
    {
        scoreText.gameObject.SetActive(false);
        taggedText.gameObject.SetActive(false);
        Time.timeScale = 0.0f;
        winGameCanvas.gameObject.SetActive(true);
        Cursor.visible = true;
        Cursor.lockState = CursorLockMode.None;
        soundEffects.clip = WinClip;
        soundEffects.Play();
    }
}
```

This is the player controller script:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.AI;

public class PlayerControllerScript : MonoBehaviour
{
    GameControllerScript gameController;

    public bool canTag = false;
    public float speed;
    public Animator playerAnimator;
    public GameObject target;

    private void Start()
    {
                                                      gameController       =
GameObject.FindGameObjectWithTag("GameController").GetComponent<GameContro
llerScript>();
    }

    void Update()
    {


    }

    public void playerMovement()
    {
        float hor = Input.GetAxis("Horizontal");
        float ver = Input.GetAxis("Vertical");
           Vector3 moveVector = new Vector3(hor, 0.0f, ver).normalized *
speed * Time.deltaTime;
        transform.Translate(moveVector, Space.Self);
        playerAnimation(hor, ver);
    }

    void playerAnimation(float horizontal, float vertical)
    {
        playerAnimator.SetFloat("HorizontalSpeed", horizontal);
        playerAnimator.SetFloat("VerticalSpeed", vertical);
    }

    public void checkPlayers()
    {
        RaycastHit hit;
```

```
                        if   (Physics.Raycast(target.transform.position,
transform.TransformDirection(Vector3.forward) * 10f, out hit))
        {
            if (hit.collider.gameObject.CompareTag("AI"))
            {
                                   Debug.DrawRay(target.transform.position,
transform.TransformDirection(Vector3.forward)       *       hit.distance,
Color.yellow);
                hit.collider.gameObject.GetComponent<AIScript>().tagAI();

            }
            else
            {
                                   Debug.DrawRay(target.transform.position,
transform.TransformDirection(Vector3.forward) * 10f, Color.white);
            }
        }
        else
        {
                                   Debug.DrawRay(target.transform.position,
transform.TransformDirection(Vector3.forward) * 10f, Color.white);
        }
    }

    private void OnTriggerStay(Collider other)
    {
        if(other.CompareTag("FlagPost"))
        {
            GameObject[] AIs = GameObject.FindGameObjectsWithTag("AI");
            foreach(GameObject ai in AIs)
            {
                ai.GetComponent<AIScript>().stopPlaying();
            }
        }
        gameController.updateScore();
    }
```

```
}
```

And this is the Base Class for the Artificial Intelligence Behavior Tree :

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;

public class AIBase : StateMachineBehaviour
{
    public GameObject player;
    public GameObject ai;
    public GameObject[] hidingSpots;
    public GameObject peakSpot;
    public GameObject currentHidingSpot;
    public GameObject spawnPoint;

    public Animator animationController;
    public NavMeshAgent agent;

      // OnStateEnter is called when a transition starts and the state
machine starts to evaluate this state
     override public void OnStateEnter(Animator animator, AnimatorStateInfo
stateInfo, int layerIndex)
    {
        ai = animator.gameObject;
        player = ai.GetComponent<AIScript>().getPlayer();
        animationController = ai.GetComponent<AIScript>().getAnimator();
        agent = ai.GetComponent<NavMeshAgent>();
    }
}
```