5 TAş oyunu Dokümentasyon

Yazılım Sınama Ödevi

Manisa Celal Bayar Üniversitesi

Muhammet Sezer Yıldırım – 192804014

İçindekiler

[**5 Taş Oyunu** 1](#_Toc57571830)

[**Amaç** 1](#_Toc57571831)

[**Girdiler** 1](#_Toc57571832)

[**Mantığı** 1](#_Toc57571833)

[**Program Kodu** 2](#_Toc57571834)

[**Ekran Çıktısı** 13](#_Toc57571835)

[**Test Caseler** 15](#_Toc57571836)

# **5 Taş Oyunu**

## **Amaç**

Oyuncu tahta üzerinde bulunan ana taş harici bütün taşlar üzerinde yazılı olan hamle sayısı kadar taşları harekete ettirerek mevcut taşları ana taş üzerinde toplaması gerekmektedir. Tahta üzerinde olabilecek duvarlara dikkat edilerek ve hamle sayısını göz önünde bulundurularak hareket edilmelidir.

## **Girdiler**

Material(GameObject): Anataş

Material(GameObject): Hamle taşları

seciliTopSatir, seciliTopSutun (float): Anataşın mevcut konumu

tahtaParcalarim (List<GameObject>): Tahta Parçları oluşumu için dizi

gamestate (bool): Oyun geçerlilik durumu

## **Mantığı**

Oyuncu oyun başlangıcından oyun modu tercihini yapmak zorundandır. Çoklu kullanıcı tercihinde kullancı adı girişinde bulunmalıdır.

Tekli oyun modunda oyun başladıktan sonra kullanıcı arayüzün sağ tarafında bulunan taşları oyun tahtası üzerine yerleştirmektedir. Ardından “Space” tuşu ile başlatılan oyunda bütün hamle taşları anataş üzerinde toplanmaktadır. Oyun öncesi yapılan yerleştirme işleminde “Mouse right click” ile tahta üzerine duvar(engel) eklenebilmektedir.

Çoklu oyun modunda yine aynı şekilde sağ tarafta bulunan taşlardan oyuncular kendi taşlarını tahta üzerine yerleştirmektedirler. Oyuncular her ikisi de “Space” tuşuna basarak hazır olduklarını belirttikleri takdirde oyun anataşın tahta üzerinde belirmesiyle başlamaktadır.

Her taş hamlesinde hareket ettirilen taş ile anataşın arasındaki mesafe hesaplanıp taş üzerinde gösterilmektedir. Oyun başlangıcında taşlar “Mouse Sol Click” ile tahtaya yerleştirilir. Taş hareket işlemi, taşın çevresine maksimum 1 adım olacak şekilde yapılmalıdır.

## **Program Kodu**

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class click : MonoBehaviour

{

public GameObject tahtam;

public GameObject panel;

load tahtaScript;

public GameObject trap;

GameObject selected=null;

public GameObject tasim;

public Text gameStateText;

public bool gameState = false;

float seciliTopSatir = 0, seciliTopSutun = 0;

int tasAdet = 0;

float x = 10.52F;

void Start()

{

tahtaScript = tahtam.GetComponent<load>();

}

void Update()

{

if (Input.GetMouseButtonDown(0))

{

seciliTopSatir = tasim.transform.position.x;

seciliTopSutun = tasim.transform.position.y;

Vector3 mousePos = Camera.main.ScreenToWorldPoint(Input.mousePosition);

Vector2 mousePos2D = new Vector2(mousePos.x, mousePos.y);

RaycastHit2D hit = Physics2D.Raycast(mousePos2D, Vector2.zero);

if (hit)

{

GameObject clickObject = hit.collider.gameObject;

if (clickObject.tag=="tas")

{

selected = clickObject;

}

else if (selected!=null&&clickObject.tag!="trap")

{

float mevTopX = clickObject.transform.position.x;

float mevTopY = clickObject.transform.position.y;

TextMesh metnim = selected.GetComponentInChildren<TextMesh>();

if (!gameState)

{

selected.transform.position = new Vector3(mevTopX, mevTopY, -1);

tahtaScript.tasMesafeOlcum();

}

else

{

if (int.Parse(metnim.text) > 0)

{

float seciliX = selected.transform.position.x;

float seciliY = selected.transform.position.y;

int mesafe = Convert.ToInt32((mevTopX > seciliX ? mevTopX - seciliX : seciliX - mevTopX) + (mevTopY > seciliY ? mevTopY - seciliY : seciliY - mevTopY));

Debug.Log($"({seciliTopSatir},{seciliTopSutun}) | ({mevTopX},{mevTopY}) - :{mesafe}");

if (mesafe == 1)

{

selected.transform.position = new Vector3(mevTopX, mevTopY, -1);

int mesafeHedef = Convert.ToInt32(metnim.text) - 1;

if (mesafeHedef == 0)

{

if (seciliTopSatir == mevTopX && seciliTopSutun == mevTopY)

{

selected.SetActive(false);

selected.transform.position = new Vector3(x, 6, -1);

x++;

tasAdet++;

if (tasAdet == 6)

{

foreach (GameObject item in tahtaScript.tas)

{

item.SetActive(true);

}

gameStateText.text = "Oyun Bitti!..";

gameStateText.color = Color.green;

gameState = false;

tahtaScript.tas[6].tag = "tas";

tahtaScript.tasYerlestir(true);

}

}

else

{

foreach (GameObject item in tahtaScript.tas)

{

item.SetActive(true);

}

gameStateText.text = "Oyun Bitti!..";

gameStateText.color = Color.red;

gameState = false;

tahtaScript.tas[6].tag = "tas";

}

}

metnim.text = mesafeHedef.ToString();

Debug.Log(metnim.text);

}

}

}

selected = null;

}

}

}

if (Input.GetMouseButtonDown(1))

{

Vector3 mousePos = Camera.main.ScreenToWorldPoint(Input.mousePosition);

Vector2 mousePos2D = new Vector2(mousePos.x, mousePos.y);

RaycastHit2D hit = Physics2D.Raycast(mousePos2D, Vector2.zero);

if (hit)

{

GameObject clickObject = hit.collider.gameObject;

Debug.Log(clickObject.tag + " : " + UnityEngine.Random.Range(0, 999));

if (clickObject.tag == "tahtaParca")

{

if (!gameState)

{

float mevTopX = clickObject.transform.position.x;

float mevTopY = clickObject.transform.position.y;

Instantiate(trap, new Vector3(mevTopX, mevTopY, -0.1F), Quaternion.identity);

}

}

else if(clickObject.tag=="trap")

{

Destroy(clickObject);

}

}

}

if (Input.GetKeyDown(KeyCode.Escape))

{

tahtaScript.tasYerlestir(false);

tahtaScript.finishManual();

panel.SetActive(true);

}

if (Input.GetKeyDown(KeyCode.Space))

{

bool startCheck = false;

foreach (GameObject item in tahtaScript.tas)

{

if (item.transform.position.x>10)

{

startCheck = false;

break;

}

else

{

startCheck = true;

}

}

if (startCheck)

{

x = 10.52F;

tasAdet = 0;

gameStateText.color = Color.white;

gameStateText.text = "Oyun Başladı";

gameState = true;

tahtaScript.tas[6].tag = "selected";

tahtaScript.StartManual();

}

}

}

}

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System;

using UnityEngine.UI;

public class load : MonoBehaviour

{

public GameObject tahtaParcam;

public GameObject tahtaParcam2;

public GameObject[] tas;

public TextMesh[] text;

public GameObject banner;

public List<List<int>> tahtam = new List<List<int>>();

List<List<GameObject>> tahtaParcalarim = new List<List<GameObject>>();

public float seciliTopSatir = 0, seciliTopSutun = 0;

public void Start()

{

if (Screen.fullScreen)

{

Screen.fullScreen = false;

}

}

public void StartManual()

{

tahtaParcaReset();

tahtaYerlestir();

tasMesafeOlcum();

}

public void finishManual()

{

foreach (List<GameObject> item in tahtaParcalarim)

{

foreach (GameObject item2 in item)

{

Destroy(item2);

}

}

tahtam.Clear();

foreach (GameObject item in tas)

{

item.SetActive(false);

}

banner.SetActive(false);

}

void tahtaParcaReset()

{

foreach (List<GameObject> item in tahtaParcalarim)

{

foreach (GameObject item2 in item)

{

GameObject.Destroy(item2);

}

}

tahtam.Clear();

for (int k = 0; k < 10; k++)

{

List<int> sutunlar = new List<int>();

for (int i = 0; i < 10; i++)

{

sutunlar.Add(0);

}

tahtam.Add(sutunlar);

}

}

public void tasMesafeOlcum()

{

seciliTopSatir = tas[6].transform.position.x;

seciliTopSutun = tas[6].transform.position.y;

for (int k = 0; k < tas.Length-1; k++)

{

float mevTopX = tas[k].transform.position.x;

float mevTopY = tas[k].transform.position.y;

if (mevTopX<10)

{

int mesafe = Convert.ToInt32((mevTopX > seciliTopSatir ? mevTopX - seciliTopSatir : seciliTopSatir - mevTopX) + (mevTopY > seciliTopSutun ? mevTopY - seciliTopSutun : seciliTopSutun - mevTopY));

text[k].text = mesafe.ToString();

}

}

}

void tahtaYerlestir()

{

for (int i = 0; i < tahtam.Count; i++)

{

List<GameObject> satirTahtaParcalarm = new List<GameObject>();

for (int k = 0; k < (tahtam[i]).Count; k++)

{

if (i % 2 == 0)

{

if (k % 2 == 0)

{

satirTahtaParcalarm.Add(Instantiate(tahtaParcam, new Vector3(i, k, 0), Quaternion.identity));

}

else

{

satirTahtaParcalarm.Add(Instantiate(tahtaParcam2, new Vector3(i, k, 0), Quaternion.identity));

}

}

else

{

if (k % 2 == 1)

{

satirTahtaParcalarm.Add(Instantiate(tahtaParcam, new Vector3(i, k, 0), Quaternion.identity));

}

else

{

satirTahtaParcalarm.Add(Instantiate(tahtaParcam2, new Vector3(i, k, 0), Quaternion.identity));

}

}

}

tahtaParcalarim.Add(satirTahtaParcalarm);

}

}

public void tasYerlestir(bool resetGame)

{

double x = 10.52;

for (int k = 0; k < 7; k++)

{

tas[k].transform.position = new Vector3((float)x, 6, -1);

if (resetGame)

{

tas[k].SetActive(true);

}

if (k!=6)

{

text[k].text = "0";

}

x++;

}

}

}

## **Ekran Çıktısı**

## **Test Caseler**

**Test Case 1:** Oyuncunun Servera bağlanması

|  |  |
| --- | --- |
| Risk Level | Yüksek |
| Purpose | Bu test durumunda kullanıcı çoklu oyuncu modunda servera bağlanılması test edilmektedir. |
| Inputs | Kullanıcın belirlediği kullanıcı adı |
| Expected Outputs | Kullanıcının çoklu oyuncu modunda servera bağlanması |
| Pass Criterias | Tek oyuncu modunda oynanma |
| Fail Criterias | Kullanıcının çoklu oyuncu modunda servera bağlanamaması |
| Test Procedure | Test kullanıcısı, yazılımı ve local server’ı; desteklenen sistem şartlarında çalıştırarak belirtilen girdiler ile testi gerçekleştirmelidir. Test sonrasında test kullanıcısı sonucu pass/fail olarak nedenleri ile birlikte raporlamalıdır. |

**Test Case 2:** Oyuncunun Mouse yordamıyla taşları hareket ettirmesi

|  |  |
| --- | --- |
| Risk Level | Yüksek |
| Purpose | Bu test durumunda taşları yerleştirmek için Mouse kullanılması durumu test edilmektedir. |
| Inputs | Dış donanım birimlerinden olan Mouse aygıtı |
| Expected Outputs | Taşların Mouse ile hareketinin sağlanması |
| Pass Criterias | Taşların klavye tuşları ile hareketinin sağlanması |
| Fail Criterias | Taşların Mouse ile hareketinin sağlanamaması |
| Test Procedure | Test kullanıcısı, yazılımı; desteklenen sistem şartlarında çalıştırarak belirtilen girdiler ile testi gerçekleştirmelidir. Test sonrasında test kullanıcısı sonucu pass/fail olarak nedenleri ile birlikte raporlamalıdır. |

**Test Case 3:** Taşların ana taşa olan mesafesinin ölçümü

|  |  |
| --- | --- |
| Risk Level | Orta |
| Purpose | Bu test durumunda kullanıcın tahta üzerine yerleştirdiği ve aynı zamanda hareket ettirdikten sonra üzerindeki belirteç ile ana taşa olan uzaklığının doğru hesaplanıp gösterilmesi durumu test edilmektedir. |
| Inputs | Taşları tahtaya yerleştirmek veya tahta üzerindeki taşları hareket ettirmek |
| Expected Outputs | Taşların ana taşa olan mesafesinin doğru hesaplanıp üzerindeki belirteç ile gösterilmesi |
| Pass Criterias | - |
| Fail Criterias | Taşların üzerindeki belirteçte ana taşa olan uzaklığının yanlış hesaplanması veya gösterilememesi |
| Test Procedure | Test kullanıcısı, yazılımı ; desteklenen sistem şartlarında çalıştırarak belirtilen girdiler ile testi gerçekleştirmelidir. Test sonrasında test kullanıcısı sonucu pass/fail olarak nedenleri ile birlikte raporlamalıdır. |