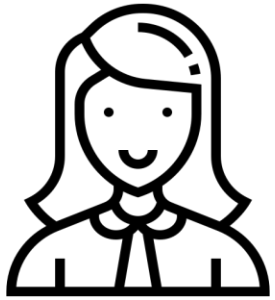
DECODE YOUR DESTINY

JULIA EGYED

Rolle

Programmiererin



Projektleiterin

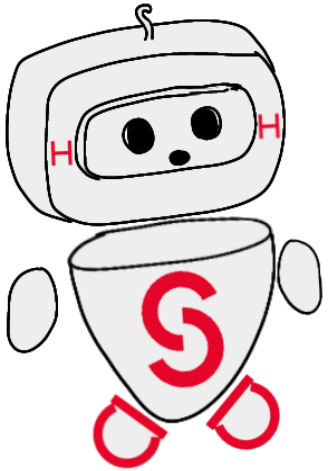
Designerin

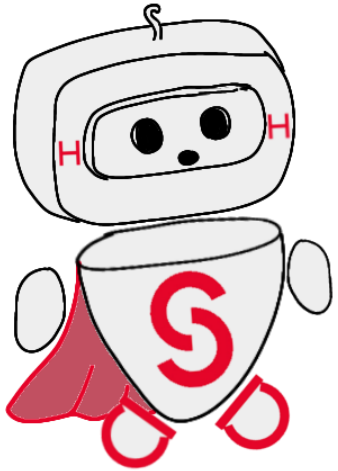
WERBECLIP HSD

Planung – Storyboard (1) & Farbkonzept

* mit ChatGPT generiert

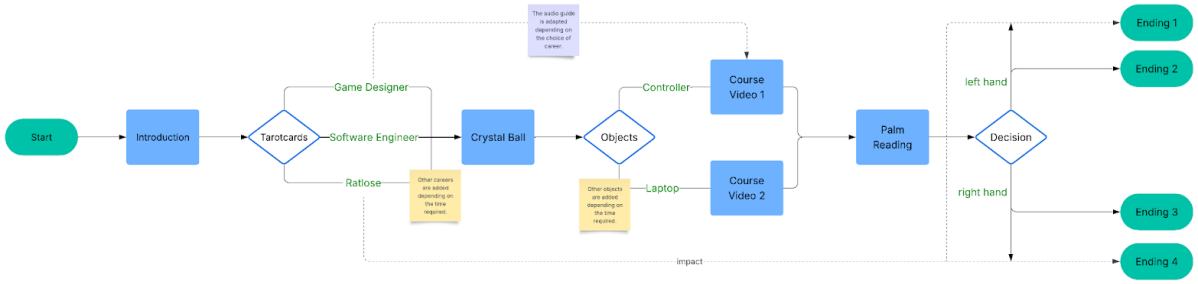
Ideen - Maskottchen





WAHRSAGERIN

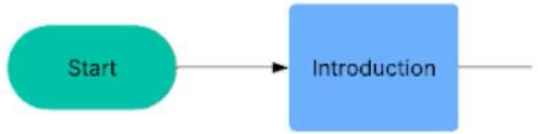
Planung – Flussdiagramm & Skript

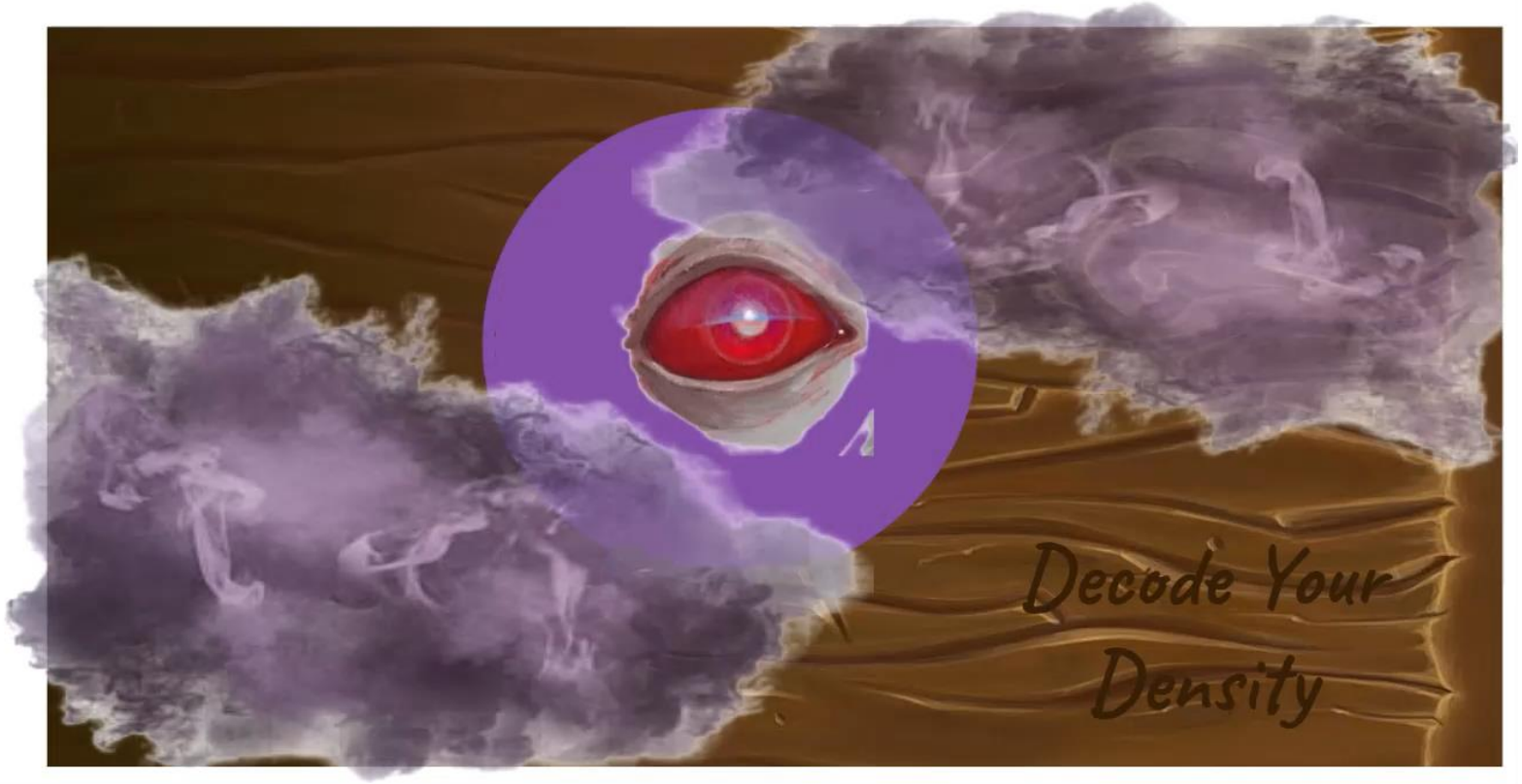


Hinweis: Je nach noch verfügbarer Zeit können wir weitere Verzweigungen/Interaktionen einbauen

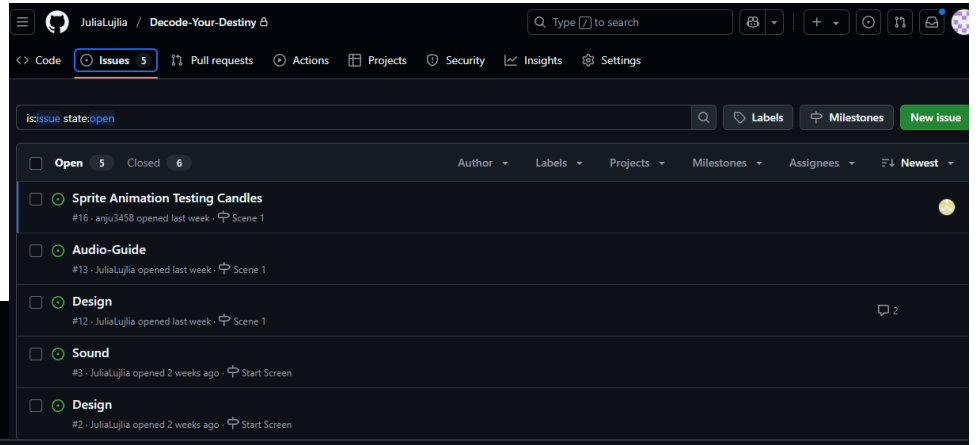
Ablauf	Beschreibung	Werkzeuge	Interaktion
Szene 1: Startscreen	Kristallkugel in der Mitte des Screens Ein Auge ist in der Kugel und schaut hindurch Restlicher Bildschirm mit Nebel/Rauch Nebel bildet die Buchstaben DECODE YOUR DESTINY Übergang: Nebel löst sich auf	Start Button, Sounds	User kann Game starten
Szene 2: Begrüßung und Einführung	Ein Tisch mit der Kristallkugel erscheint Kerzen, Kristalle und weitere typische Wahrsager Elemente sind als Deko da Tarotkarte sind in der Ecke sichtbar Hände von alter, ungepflegten Wahrsagerin sind zu sehen Die Wahrsagerin begrüßt ihnen und erklärt, dass sie nun die Zukunft vom User vorhersagt Dafür legt sie die Tarotkarten "vor" den User Jede Karte repräsentiert ein Berufsfeld (der mit dem Studiengang MI möglich ist, User weiß das aber nicht) oder die Karte "Ratlose*r" Sie fordert den User auf eine Karte zu wählen Übergang: Zoom in die Kristallkugel	Audio-Guide, Tarotkarten als Buttons, Hilfs- & Restart Button	User kann eine Karte wählen
Szene 3: Blick in die Kristallkugel	In der Kristallkugel befinden sich Objekte, die Kurse der HSD repräsentieren z.B. Laptop, Kontroller, Kamera, Das/die Objekte leuchtet/n auf, welche/s zum ausgewählten Beruf passt/en (beim Ratlosen wird nichts markiert) Wahrsagerin erklärt, welche/s Objekt/e am besten zum ausgewählten Beruf passt Bevor sie die Zukunft des Users vorhersagen kann, bittet sie den User so viele Objekte wie möglich zu betrachten Nebelwolken fliegen in der Kugel rum	Audio-Guide, Hilfs- & Restart Button	
Szene 4: Einblick in die Kurse	User interagiert mit einem Objekt, Kursvideo läuft ab Wie ein Start/Stop Button, nur dass das Video erst läuft, wenn mit dem Objekt interagiert wird Der User muss mit den Objekten interagieren, bis alle Videos abgespielt wurden Grafisch wird eine Prozessbar angezeigt Solange die Prozessbar nicht voll ist, kann die Wahrsagerin keine vorhersage machen: Wahrsagerin gibt immer Zwischenkommentare ab In den Videos werden Kurse der HSD dargestellt, die eine Verbindung mit dem Objekt haben Übergang: Rauszoomen aus der Kugel	Sounds, Interaktive Objekte, Hilfs- & Restart Button	User kann die Objekte berühren/drehen
Szene 5: Die Entscheidung	Damit die Wahrsagerin nun eine Vorhersage treffen kann, bittet sie den User seine Hand in einer ihrer Hände zu legen Sie schreckt ihre Hände nach "vorne" Option 1: linke Hand - User findet die Kursangebote interessant Option 2: rechts Hand - User findet die Kursangebote uninteressant mysteriöser Nebel sammelt sich an den Händen	Audio-Guide, Hände als Buttons, Hilfs- & Restart Button	User kann sich für eine Hand entscheiden
Szene 6: Abschluss	Option 1: Möglichkeit 1: Sie sieht in der Kugel den User an der HSD Medieninformatik studieren Möglichkeit 2: Sie sieht den User auch an der HSD Medieninformatik studieren, erzählt aber noch dass man durch die Vielfalt dann schon einen Berufsweg finden wird Option 2: Möglichkeit 1: Sie hat noch keine klare Vision bekommen, schlägt dem User vor weitere Projekte des Fachbereichs Medien anzuschauen Medieninformatik kann einen aber den gewünschten Beruf ermöglichen, ob ei nun an der HSD ist, ist unklar Möglichkeit 2: (falls der/die Ratlose gewählt wurde) Sie rät dem User sich in anderen Bereichen der HSD umzusehen, vielleicht könnte ein anderer Studiengang an der HSD den User interessieren Anmerkung: direkte Verweis darauf, dass man sich am Tag der offenen Tür an der HSD befindet	Audio-Guide	

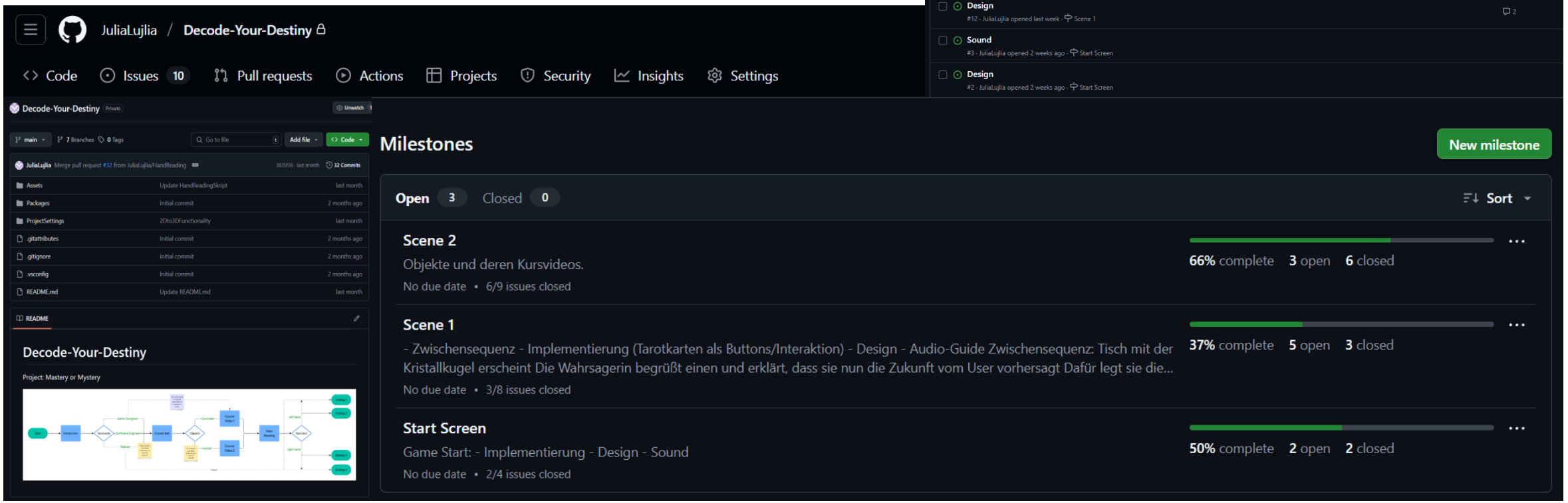
Planung – Storyboard (2)





Planung – 2 x GitHub & Unity





Ideen – Table Design

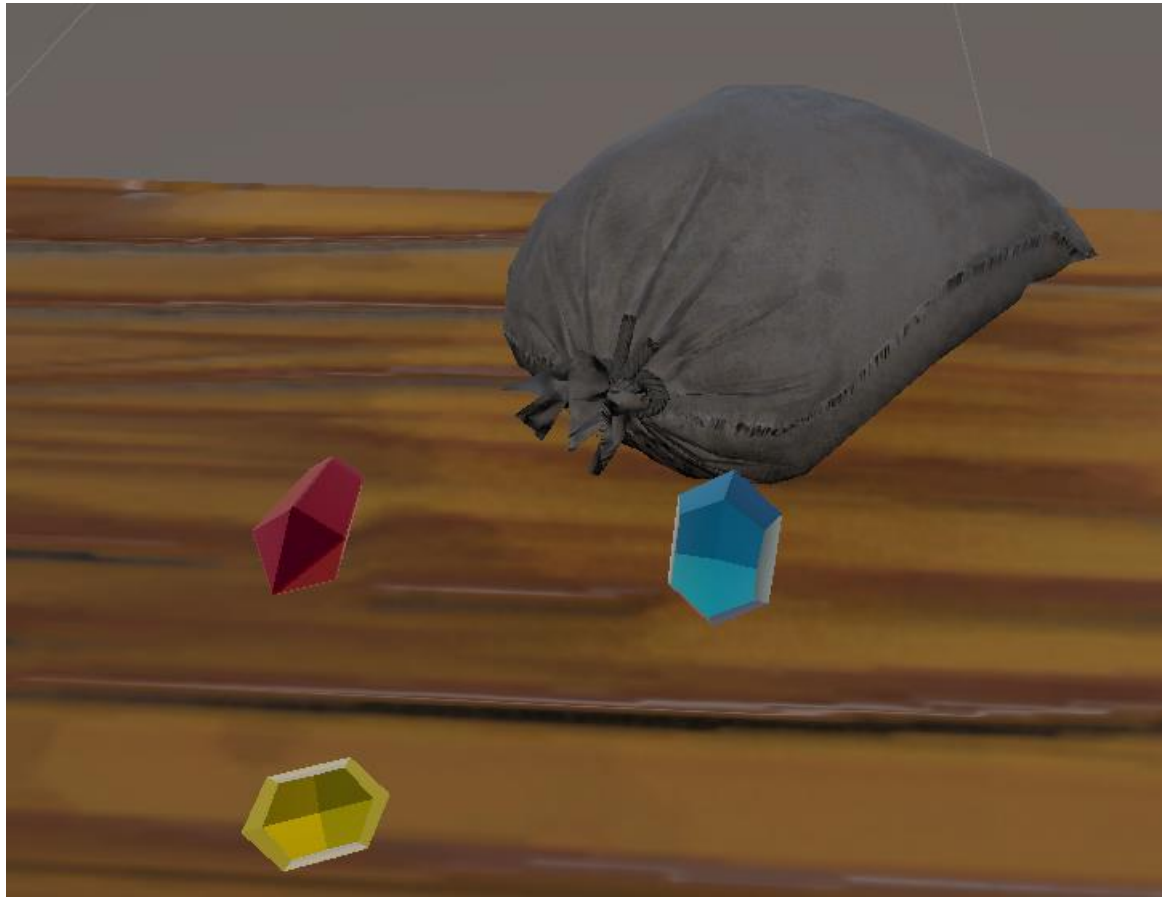
Ideen – Hand Design

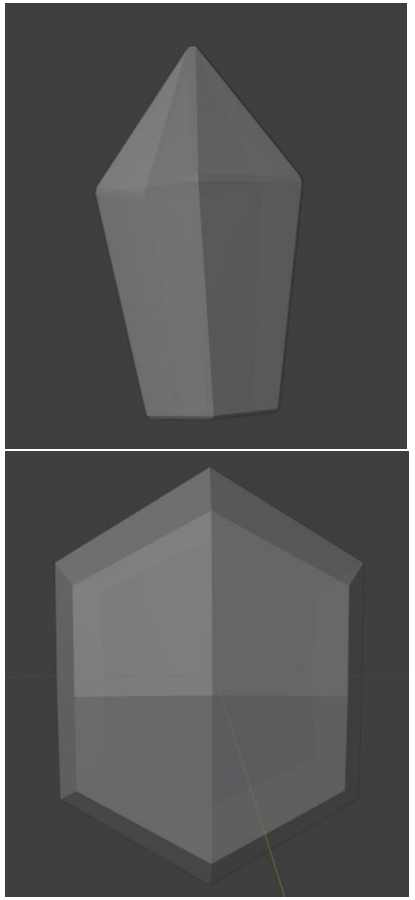
Idee

ChatGPT

Kombination

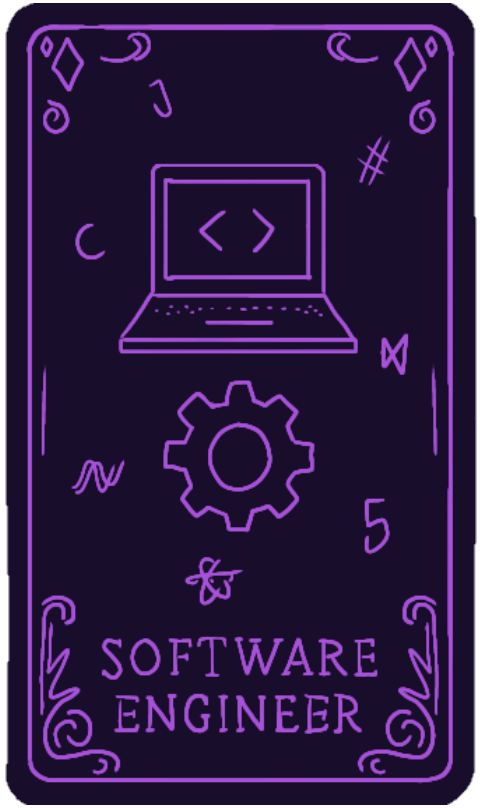
Ideen – Taschen Design





+ Kristalle in Blender verändert

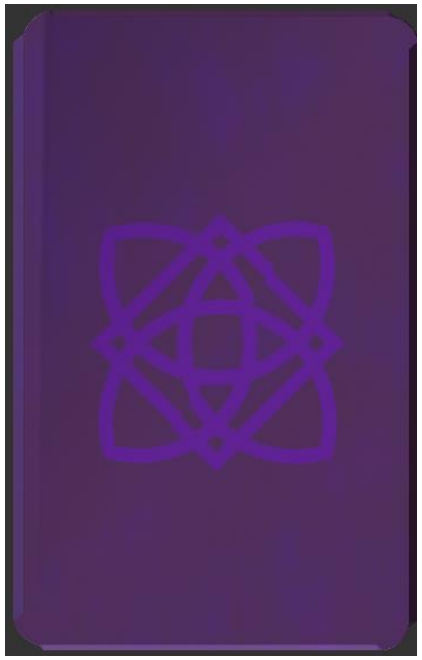
Ideen – Tarotkarten Design

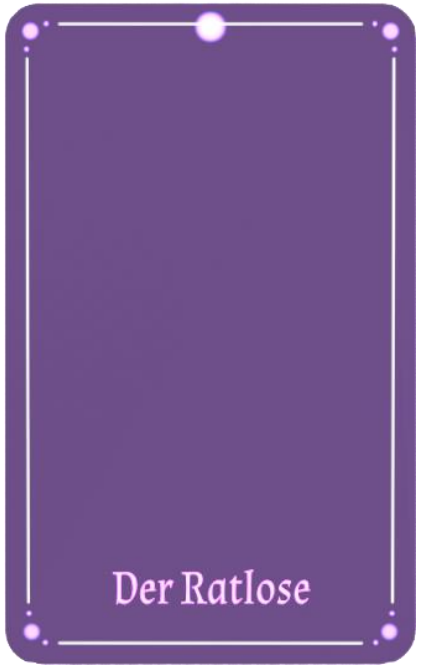


Idee

ChatGPT

Design – Tarotkarten



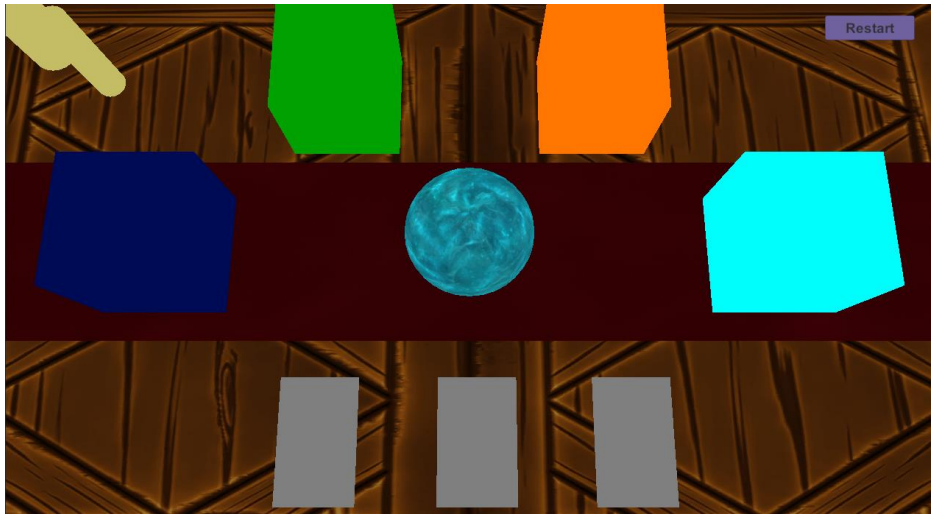


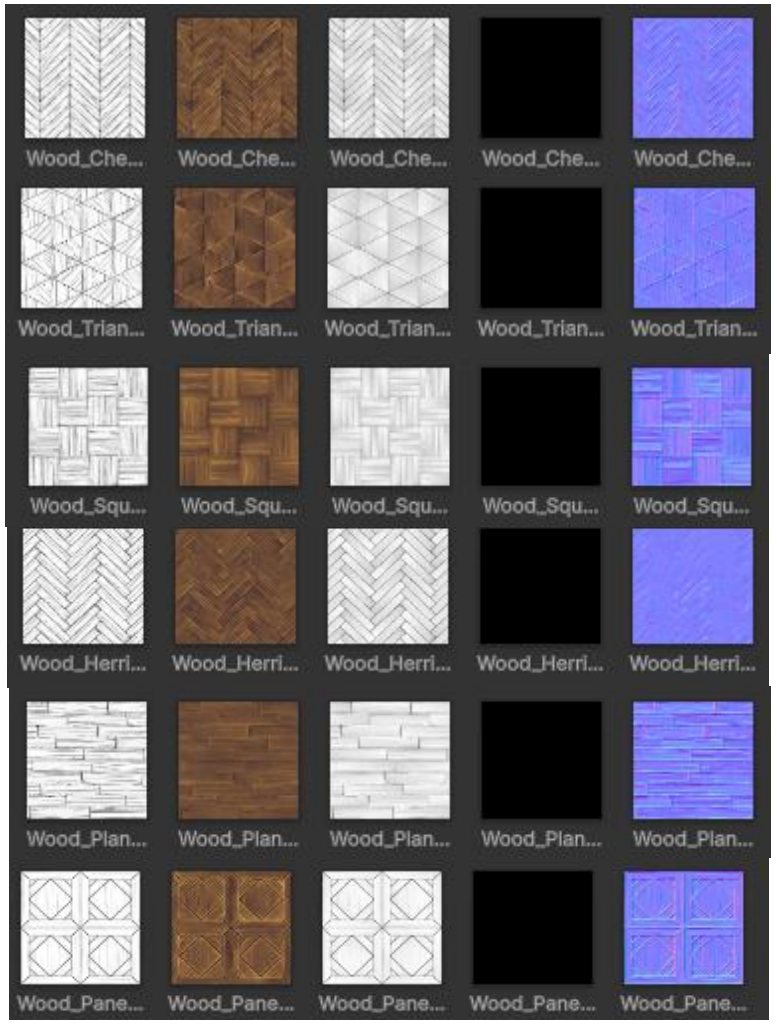
Design – Restart Button

Idee

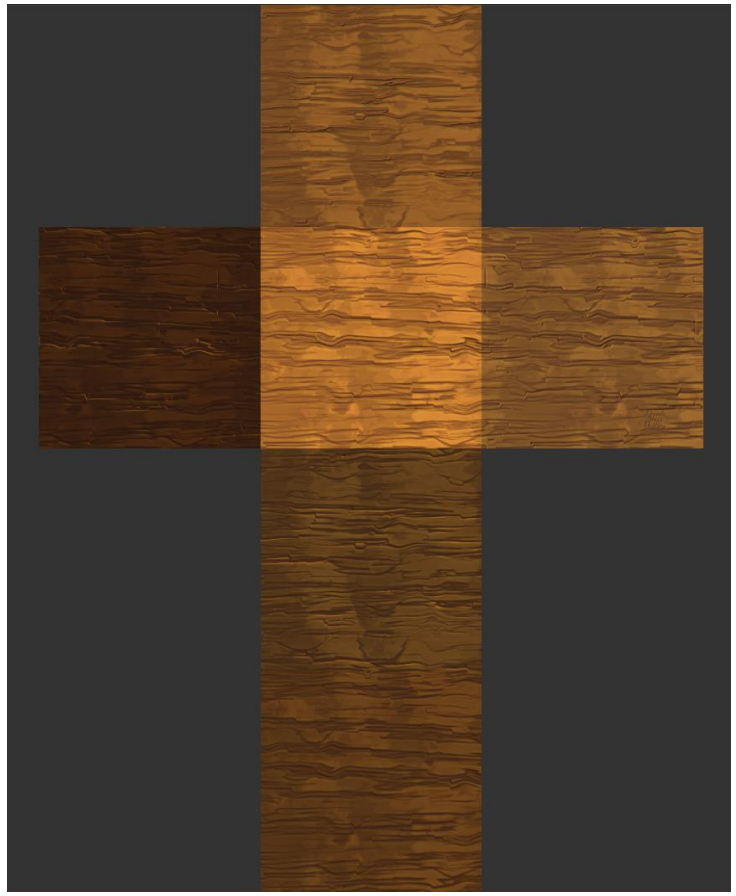
ChatGPT

Design – Table

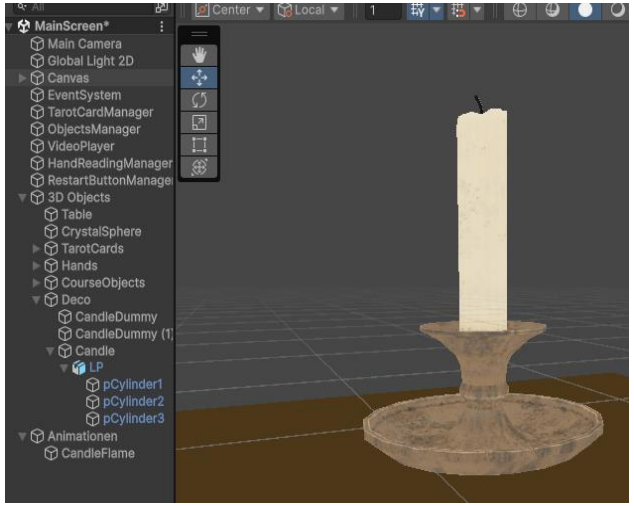


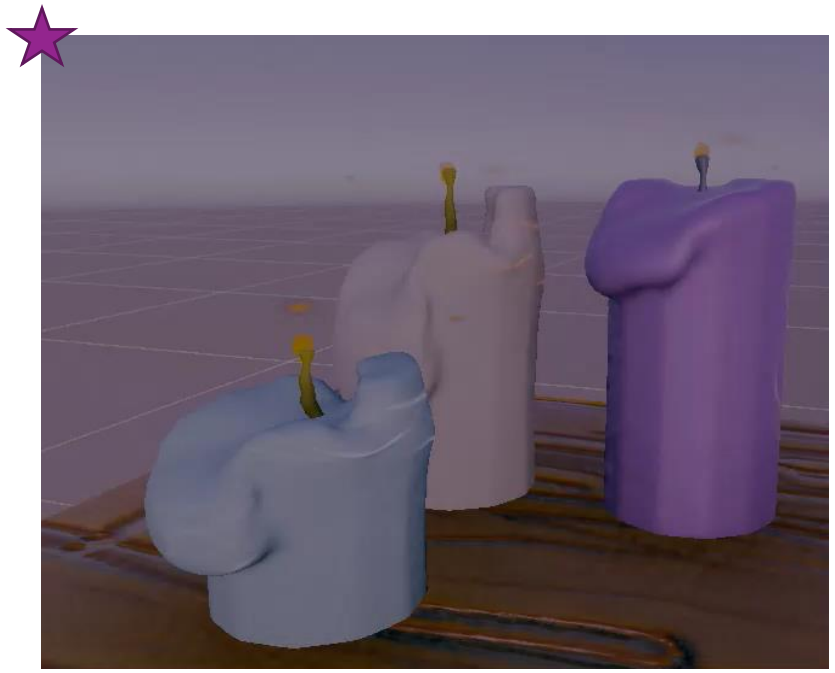


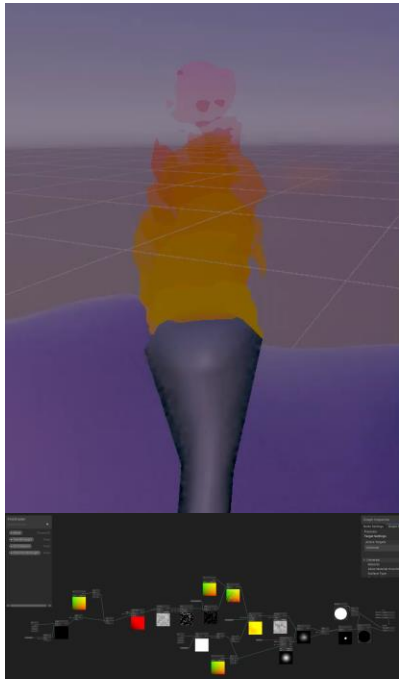
Design – Table



Design, Partikelsystem & Shader – Kerzen



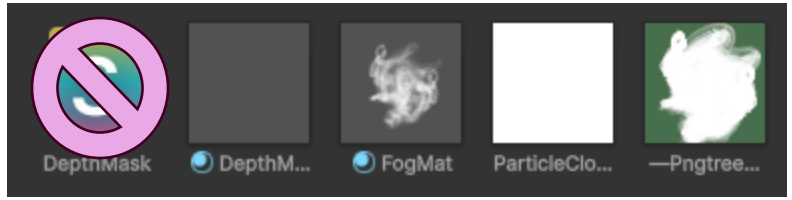


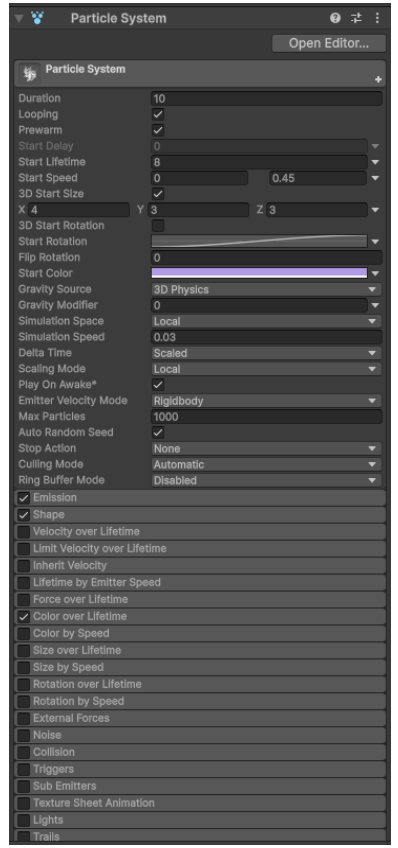


Version 1

Version 2

Partikelsystem – Nebel





+ DontDestroy Skript

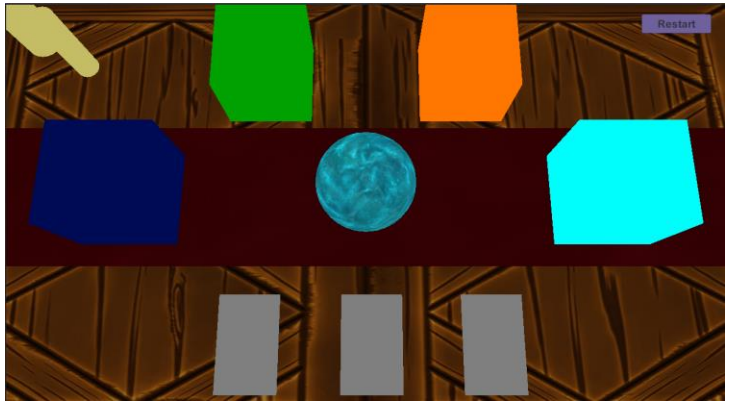
weitere Mitwirkungen am Design

- Lichter (nicht: flackern, Kristalllicht)
- Kamera
- Platzierung
- Start & Endszene (Schriftart + Animation/Skript)

Programmierung

177	14	9	14	20	122 + Andreas	28	264	12	14	10	18	84

Zusätzliches



- Blocking
- Settings, Anti-Aliasing
- Merge (Konflikte)
- Audio + Videos eingebaut
- Soundeffekt (Kerzen)

Was habe ich gelernt?

- Unity (C#, Partikelsystem, Shader, 2D vs 3D, …)
- Blender (einfache Modellierungen)
- Substance 3D Painter

Welche Baustellen sind mir aufgefallen?

- Projektplanung (schriftliches Festhalten, …)

  Psychologie im Projektmanagement: Projektleitung in komplexen Organisationen | SpringerLink

  Leitfaden Projektmanagement

Danke

Programmierung


```
{
    public AudioSource[] audioSources;

    // Flags, um zu kontrollieren, welcher Guide abgespielt werden soll
    public bool playIntro;
    public bool playAfterCardSelected;
    public bool playAfterAllObjects;
    public bool playAfterRatloserRight;
    public bool playAfterRatloserLeft;
    public bool playAfterJobRight;
    public bool playAfterJobLeft;

    private bool[] wasPlayed; // Um Dopplungen zu vermeiden

    // Collider help var

    public bool firstAudioFinished;
    private bool waitingForFirstAudio = false;
    public bool secondAudioFinished;
    private bool waitingForSecondAudio = false;
    public bool thirdAudioFinished;
    private bool waitingForThirdAudio = false;

    private bool waitingForEndAudio = false;
    private int endAudioIndex = -1;
    private bool endSceneShown = false;

    // End Screen
    public GameObject endScreen;
    public Vector3 startPosition;
    public Vector3 targetPosition;
    public float moveSpeed = 2f;
    private bool animateEndScreen = false;
    private CanvasGroup endScreenCanvasGroup;



    Unity-Nachricht | 0 Verweise
    void Start()
    {
        // Track
        wasPlayed = new bool[audioSources.Length];

        // End Screen
        if (endScreen != null)
        {
            endScreen.transform.localPosition = startPosition;
            endScreen.SetActive(false);
        }
    }

    Unity-Nachricht | 0 Verweise
    void Update()
    {
        // INTRO GUIDE
        if (playIntro && !wasPlayed[0])
        {
            PlayGuide(0);
            waitingForFirstAudio = true;
            wasPlayed[0] = true;
        }

        if (waitingForFirstAudio && !audioSources[0].isPlaying)
        {
            firstAudioFinished = true;
            waitingForFirstAudio = false;
            Debug.Log("AudioGuide 0 finished.");
        }

        // After Card selected
```

Programmierung


```
using UnityEngine;

Unity-Skript (3 Objektverweise) | 0 Verweise
public class DerRatloseButtonScript : MonoBehaviour
{
    public TarotCardScript manager;

    Unity-Nachricht | 0 Verweise
    private void OnMouseUpAsButton()
    {
        if (manager != null)
        {
            manager.OnTarotCardClicked(gameObject);
        }
    }
}
```

Programmierung


```
using UnityEngine;

Unity-Skript (1 Objektverweis) | 0 Verweise
public class FogScript : MonoBehaviour
{
    Unity-Nachricht | 0 Verweise
    void Awake()
    {
        DontDestroyOnLoad(gameObject);
    }
}
```

Programmierung


```
1   using UnityEngine;
2
    Unity-Skript (4 Objektverweise) | 0 Verweise
3   public class HandButtonScript : MonoBehaviour
4   {
5       public HandReadingScript handScript;
6
        Unity-Nachricht | 0 Verweise
7       void Start()
8       {
9           Debug.Log("HandButtonScript");
10      }
11
        Unity-Nachricht | 0 Verweise
12      void OnMouseDown()
13      {
14          if (handScript != null)
15          {
16              handScript.HandDescion(gameObject.name);
17              Debug.Log(gameObject.name);
18          }
19      }
20  }
```

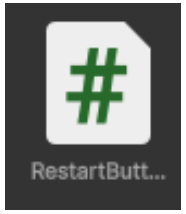
Programmierung


```
public GameObject leftHand;
public GameObject rightHand;

public TarotCardScript tarotCardScript;
public Animator handsAnimator;

private bool handCollidersActivated = false;
private bool outroPlayed = false;

1 Verweis
public void PlayOutroAnimation()
{
    if (!outroPlayed)
    {
        handsAnimator.SetTrigger("PlayOutroAnimation");
        outroPlayed = true;
        Debug.Log("Outro animation triggered.");

        TriggerChooseLeftDelayed();
    }
}

0 Verweise
public void MoveHands()
{
    Debug.Log("Hand Reading");

    leftHand.transform.position += new Vector3(0, 0, 0);
    rightHand.transform.position += new Vector3(0, 0, 0);

    // Acctivate Collider
    //leftHand.GetComponent<Collider>().enabled = true;
    //rightHand.GetComponent<Collider>().enabled = true;
}

Unity-Nachricht | 0 Verweise
void Update()
{
    if (!handCollidersActivated)
    {
        AudioGuideScript guide = FindObjectOfType<AudioGuideScript>();
        if (guide != null && guide.thirdAudioFinished)
        {
            leftHand.GetComponent<Collider>().enabled = true;
            rightHand.GetComponent<Collider>().enabled = true;

            handCollidersActivated = true;
            Debug.Log("Hand-Collider activated after Audio.");
        }
    }
}

1 Verweis
public void HandDescion(string handName)
{
    Debug.Log("HandDescion Script");

    string selectedTarotCard = tarotCardScript.GetSelectedTarotCard();
    Debug.Log("Hand: " + handName);
    Debug.Log("Tarot card: " + selectedTarotCard);

    if (handName == "LeftHand")
    {
        TriggerChooseLeftDelayed();
    }
    else if (handName == "RightHand")
    {
        handsAnimator.SetTrigger("PlayChooseRight");
    }
```

Programmierung


```
using UnityEngine;
using UnityEngine.Video;
using UnityEngine.UI;
using System;

Unity-Skript (1 Objektverweis) | 1 Verweis
public class ObjectsScript : MonoBehaviour
{
    public GameObject[] choiceObjects;
    // Skripts (previous scene script and upcoming scene script)
    public TarotCardScript tarotCardScript;
    public HandReadingScript handReadingScript;

    // Video
    public VideoPlayer videoPlayer;
    public VideoClip[] videoClips;
    public GameObject videoScreen;

    // Testing Audio Version
    public AudioSource[] audioSources;

    // Count Course Videos, for EventTrigger: Hand Reading
    private int requiredClicks = 0;
    private int clickedCount = 0;
    private bool[] hasBeenWatched;

    // Start Event after last Video
    private bool allButtonsClicked = false;
    private int lastVideoPlayedIndex = -1;

    private bool objectCollidersActived = false;


    // Start is called once before the first execution of Update after the MonoBehaviour is created
    Unity-Nachricht | 0 Verweise
    void Start()
    {
        // Initialize Watched List
        hasBeenWatched = new bool[choiceObjects.Length];

        foreach (GameObject obj in choiceObjects)
        {
            // Deactivate all Objects
            obj.GetComponent<Collider>().enabled = false;
            obj.SetActive(false);
            videoScreen.SetActive(false);
        }

        videoPlayer.audioOutputMode = VideoAudioOutputMode.None;

        // Finished Video
        videoPlayer.loopPointReached += OnVideoFinished;
    }

    Unity-Nachricht | 0 Verweise
    void Update()
    {
        // Wait till Audio Guide is finished
        if (!objectCollidersActived)
        {
            AudioGuideScript guide = FindObjectOfType<AudioGuideScript>();
            if (guide != null && guide.secondAudioFinished)
            {
                EnableChoiceObjectColliders();
                objectCollidersActived = true;
            }

        }
    }

    // Show Objectes based on the previous selected Tarot Card
```

Programmierung




```
1   using UnityEngine;
2   using UnityEngine.SceneManagement;
3
    Unity-Skript (1 Objektverweis) | 0 Verweise
4   public class RestartButtonScript : MonoBehaviour
5   {
6       // Start is called once before the first execution of Update after the MonoBehaviour is created
        0 Verweise
7       public void RestartGame()
8       {
9           // Load Start Scene
10          SceneManager.LoadScene("StartScreen");
11      }
12  }
13
```

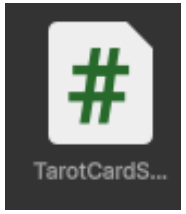
Programmierung


```
1   using UnityEngine;
2   using UnityEngine.SceneManagement;
3
        Unity-Skript (1 Objektverweis) | 0 Verweise
4   public class StartGameScript : MonoBehaviour
5   {
            0 Verweise
6       public void StartGame()
7       {
8           SceneManager.LoadSceneAsync("MainScreen");
9       }
10  }
```

Programmierung


```
using UnityEngine;

Unity-Skript (1 Objektverweis) | 0 Verweise
public class SystemManagerScript : MonoBehaviour
{
    Unity-Nachricht | 0 Verweise
    void Awake()
    {
        DontDestroyOnLoad(gameObject);  // damit es in MainScreen auch funktioniert
    }

    Unity-Nachricht | 0 Verweise
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            Application.Quit();
            Debug.Log("Quit Game");
        }
    }
}
```

Programmierung




```
4
    Unity-Skript (1 Objektverweis) | 5 Verweise
5   public class TarotCardScript : MonoBehaviour
6   {
7       // public Button[] TarotCardButtons;
8       public GameObject[] TarotCards;
9       private string selectedTarotCard;
10
11      // Event Trigger
12      [System.Serializable]
        1 Verweis
13      public class TarotCardSelectedEvent : UnityEvent<string> { }
14      public TarotCardSelectedEvent OnTarotCardSelectedEvent;
15
16      private bool collidersActived = false;
17
        Unity-Nachricht | 0 Verweise
18      private void Start()
19      {
20          UnityEngine.Object.FindFirstObjectByType<AudioGuideScript>().playIntro = true;
21
22          foreach (GameObject card in TarotCards)
23          {
24              card.GetComponent<Collider>().enabled = false;
25          }
26      }
        Unity-Nachricht | 0 Verweise
27      void Update()
28      {
29          // Wait till Audio Guide is finished
30          if (!collidersActived)
31          {
32              AudioGuideScript guide = FindObjectOfType<AudioGuideScript>();
33              if (guide != null && guide.firstAudioFinished)
34              {
35                  EnableChoiceObjectColliders();
36                  collidersActived = true;
37              }
38          }
39      }
40
        1 Verweis
41      void EnableChoiceObjectColliders()
42      {
43          foreach (GameObject card in TarotCards)
44          {
45              if (card.activeSelf)
46              {
47                  Collider col = card.GetComponent<Collider>();
48                  if (col != null)
49                      col.enabled = true;
50              }
51          }
52          Debug.Log("Collider activated after Audio.");
53      }
54
        3 Verweise
55      public void OnTarotCardClicked(GameObject clickedCard)
56      {
57          // Save selected Tarot Card
58          selectedTarotCard = clickedCard.name;
59          Debug.Log("Selected Tarot Card: " + selectedTarotCard);
60
61          // Deactivate Collider
62          foreach (GameObject card in TarotCards)
63          {
64              Collider cardCol = card.GetComponent<Collider>();
65              if (cardCol != null)
66              {
67                  cardCol.enabled = false;
68              }
```