

Implementierte PoCs im funktionalen Prototyp

Type des Projektrisikos & PoCs Benennung	Beschreibung der PoC Funktionalität	Exit-Kriterien	Fail-Kriterien	Fallbacks
• technisch, architekturell Laden von Szenen	<ul style="list-style-type: none"> TidyShinyFireworks besitzt aktuell 3 Szenen, welche eine (Teil-)Lösung für die Domäne darstellen. Szenen werden über deren Namen oder einen Index, welcher zum Build-Zeitpunkt festgelegt wird, referenziert. Der SzeneManager von Unity lädt die Szene im Hintergrund. Währenddessen wird eine Animation abgespielt, um einen saubereren Übergang zur nächsten Szene darzustellen. Erst wenn die Szene geladen wurde, wird der zweite Teil der Animation abgespielt. 	<ul style="list-style-type: none"> Die nächste Szene wird vom SzeneManager gefunden, die Animation wird gestartet und wird beendet, wenn die neue Szene fertig geladen wurde. 	<ul style="list-style-type: none"> Es wurde kein Int oder String übergeben oder der Szenenindex bzw. Szenenname ist nicht bekannt. 	<ul style="list-style-type: none"> • KEIN FALLBACK MÖGLICH => Exceptions werden geworfen
• technisch QR Scanning	<ul style="list-style-type: none"> In der zweiten Szene wird mittels Unitys WebCamTexture das Bildmaterial der Kamera aufgezeichnet. Dessen Bildmaterial wird von BarcodeReader.Decode() verarbeitet. Diese Verarbeitung findet auf einem separaten Thread statt, um die Performance von WebCamTexture nicht einzuschränken. 	<ul style="list-style-type: none"> Der Inhalt des QR Codes wird erkannt und in einem Textelement der UI gesichert. 	<ul style="list-style-type: none"> Es kann kein neuer Thread erstellt werden oder es wird kein QR Code erkannt. 	<ul style="list-style-type: none"> • Zukunftsfeature: Der Nutzer öffnet ein Menü, welches im alle bereits auf dem Smartphone zur Verfügung stehenden AssetBundles (Feuerwerke) anbietet, um die AR Szene zu starten.
• technisch, architekturell Downloaden von AssetBundles	<ul style="list-style-type: none"> Aus dem Resultat des Inhaltes und einem Google Drive Präfix wird mittels UnityWebRequest ein Assetbundle heruntergeladen, falls dieses noch nicht vorher geladen wurde und wird unter Application.persistentDataPath/3Modelle/ gesichert. 	<ul style="list-style-type: none"> Der Download wird über einen Button gestartet und erfolgt im Hintergrund. 	<ul style="list-style-type: none"> Der Download kann nicht gestartet bzw. beendet werden oder der Downloadpfad ist ungültig. 	<ul style="list-style-type: none"> • Dem Nutzer wird im Textfeld der UI ein Fehler angezeigt mit weiteren Informationen.
• technisch, architekturell Platzieren des Feuerwerks in AR	<ul style="list-style-type: none"> Wenn ein Feuerwerksmodell im Speicher ausgewählt wurde (neu geladen, Vorhandensein erkannt oder zukünftig aus Menü), kann die AR Szene geladen werden. Erst wenn der Platzieren-Button gedrückt wurde, wird das Assetbundle zur Laufzeit in ein Prefab verwandelt, welches dem TrackedImageManager übergeben wird. Wenn nun ein Referenzbild (hier ein QR Code) erkannt wird, wird dort in der AR Szene das Objekt vom Manager platziert. 	<ul style="list-style-type: none"> Das AssetBundle wurde aus dem Speicher geladen und in der Szene platziert. 	<ul style="list-style-type: none"> Das Referenzbild wird in der AR Szene nicht erkannt, sodass kein Modell platziert wird. 	<ul style="list-style-type: none"> • Der Nutzer kann wieder in die QR Scanning Szene zurückspringen und es mit einem anderen Modell probieren. SONST KEIN FALLBACK
• technisch Starten des Feuerwerks	<ul style="list-style-type: none"> Das ParticleSystem des in der Szene platzierten Feuerwerks wird einmalig abgespielt. 	<ul style="list-style-type: none"> Das ParticleSystem wird abgespielt und wird automatisch beendet. 	<ul style="list-style-type: none"> Das Modell besitzt nicht die vorgeschriebene Struktur und das ParticleSystem wird nicht gestartet 	<ul style="list-style-type: none"> • Der Nutzer versucht das Gleiche mit einem anderen Modell.
• technisch "Firework Visualisation and Sound"	<ul style="list-style-type: none"> Die VFX-Ausführung wird akustisch begleitet. 	<ul style="list-style-type: none"> Neben den VFX wird zum richtigen Zeitpunkt SFX ausgeführt 	<ul style="list-style-type: none"> (a) SFX werden nicht oder zeitlich versetzt abgespielt. 	<ul style="list-style-type: none"> • (a) Der Nutzer bricht dem Vorgang ab, um das nächste Feuerwerk zu platzieren.
• technisch, kommunikativ, kompetenzorientiert	<ul style="list-style-type: none"> Gerade Events wie Feuerwerk möchte man in Gesellschaft erleben. Deshalb soll mittels Multi-User-AR dieses Gefühl bewahrt werden. Es werden Standort und Sichtwinkel der einzelnen Geräte benötigt, um 3D Modelle für alle korrekt zu platzieren und die VFX/SFX auszulösen. 	<ul style="list-style-type: none"> Das Feuerwerk wird korrekt für alle teilnehmenden Geräte angezeigt. 	<ul style="list-style-type: none"> Zuschauergeräte erhalten falsche oder keine Informationen. 	<ul style="list-style-type: none"> • Eine erneute Anfrage der Darstellungsinformationen an den Host der AR-Show.

TidyShinyFireworks



Kevin Repke
Paul Johne

Audit 4



Einleitung



Zusätzliche
Artefakte



funktionaler
Prototyp



Finish

TidyShinyFireworks

Inhaltsverzeichnis 1/2

- Finaler Projektplan
- Überarbeitete Proof of Concepts
- Cognitive Walkthrough
- Vision-Exposé
- Vorstellung des funktionalen Prototypen
 - Levelloader
 - AndroidCam + Torch
 - QR Scanner
 - Model mit Particle Systems
 - Asset Bundle Downloader
 - AR Logic

- heute geplant Vorstellung ist die Vorstellung des funktionalen Prototypen und der zusätzlichen Artefakte
- gestartet wird mit dem Projektplan
- dann gehen wir kurz auf die Artefakte ein. Wegen der Zeit, bitte ich Sie, diese im Nachhinein genauer anzuschauen
- die Vorstellung des funktionalen Prototypen anhand eines Videos, sowie der wichtigsten Skripte, die die Funktionen ermöglichen



Einleitung



Zusätzliche
Artefakte



funktionaler
Prototype



Finish

TidyShinyFireworks

Inhaltsverzeichnis 2/2

- Plakat
- Erweiterungsmöglichkeiten für eine vollwertige Version
- Fazit und kritische Reflexion

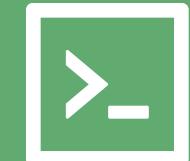
- zum Schluss präsentieren wir Ihnen das Plakat
- Erweiterungsmöglichkeiten für eine vollwertige Version
- sowie unser Fazit und unsere kritische Reflexion zum Projekt



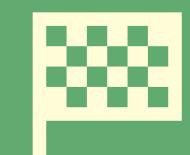
Einleitung



Zusätzliche Artefakte



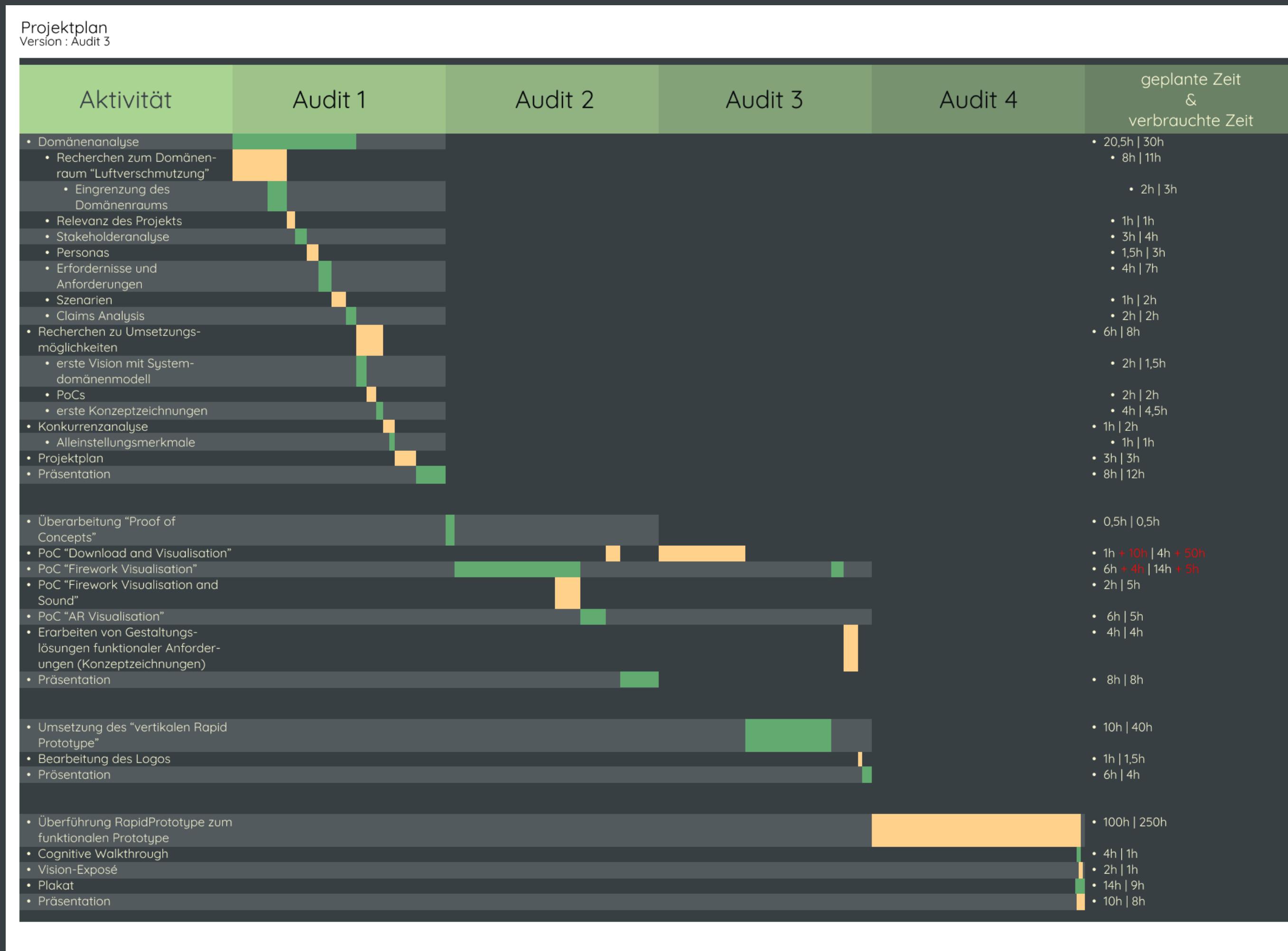
funktionaler Prototype



Finish

TidyShinyFireworks

Projektplan



- in den letzten Wochen haben wir folgende Dinge geplant und umgesetzt
 - der funktionale Prototyp hat am meisten Zeit gekostet. Darin liegt unsere Hauptaufgabe
 - danach haben wir noch zusätzliche Artefakte erstellt, um noch neben dem Prototypen Nützliches zu präsentieren
 - das Plakat und die Präsentation



Proof of Concepts

Überarbeitete Risiken 1/2

Implementierte PoCs im funktionalen Prototyp				
Typ des Projektrisikos & PoCs Benennung	Beschreibung der PoC Funktionalität	Exit-Kriterien	Fail-Kriterien	Fallbacks
• technisch, architekturell Laden von Szenen	• TidyShinyFireworks besitzt aktuell 3 Szenen, welche eine (Teil-)Lösung für die Domäne darstellen. Szenen werden über deren Namen oder einen Index, welcher zum Build-Zeitpunkt festgelegt wird, referenziert. Der SzeneManager von Unity lädt die Szene im Hintergrund. Währenddessen wird eine Animation abgespielt, um einen sauberen Übergang zur nächsten Szene darzustellen. Erst wenn die Szene geladen wurde, wird der zweite Teil der Animation abgespielt.	• Die nächste Szene wird vom SzeneManager gefunden, die Animation wird gestartet und wird beendet, wenn die neue Szene fertig geladen wurde.	• Es wurde kein Int oder String übergeben oder der Szenenindex bzw. Szenename ist nicht bekannt.	• KEIN FALLBACK MÖGLICH => Exceptions werden geworfen
• technisch QR Scanning	• In der zweiten Szene wird mittel Unitys WebCamTexture das Bildmaterial der Kamera aufgezeichnet. Dessen Bildmaterial wird von BarcodeReader.Decode() verarbeitet. Diese Verarbeitung findet auf einem separaten Thread statt, um die Performance von WebCamTexture nicht einzuschränken.	• Der Inhalt des QR Codes wird erkannt und in einem Textelement der UI gesichert.	• Es kann kein neuer Thread erstellt werden oder es wird kein QR Code erkannt.	• Zukunftsfeature: Der Nutzer öffnet ein Menü, welches ihm alle bereits auf dem Smartphone zur Verfügung stehenden AssetBundles (Feuerwerke) anbietet, um die AR Szene zu starten.
• technisch, architekturell Downloaden von AssetBundles	• Aus dem Resultat des Inhaltes und einem Google Drive Präfix wird mittels UnityWebRequest ein Assetbundle heruntergeladen, falls dieses noch nicht vorher geladen wurde und wird unter Application.persistentDataPath/3Modelle/ gesichert.	• Der Download wird über einen Button gestartet und erfolgt im Hintergrund.	• Der Download kann nicht gestartet bzw. beendet werden oder der Downloadpfad ist ungültig.	• Dem Nutzer wird im Textfeld der UI ein Fehler angezeigt mit weiteren Informationen.

- im Repo unter den Artefakten zu finden
- zur Atomarisierung des funktionalen Prototyps



- Überarbeitung der PoCs
 - neue Funktionen wurden hinzugefügt
 - nicht brauchbare oder realisierte Funktionen wurden gestrichen



Proof of Concepts

Überarbeitete Risiken 1/2



- technisch, architekturell
Platzieren des Feuerwerks in AR

- Wenn ein Feuerwerksmodell im Speicher ausgewählt wurde (neu geladen, Vorhandensein erkannt oder zukünftig aus Menü), kann die AR Szene geladen werden. Erst wenn der Platzieren-Button gedrückt wurde, wird das AssetBundle zur Laufzeit in ein Prefab verwandelt, welches dem TrackedImageManager übergeben wird. Wenn nun ein Referenzbild (hier ein QR Code) erkannt wird, wird dort in der AR Szene das Objekt vom Manager platziert.

- Das AssetBundle wurde aus dem Speicher geladen und in der Szene platziert.

- Das Referenzbild wird in der AR Szene nicht erkannt, sodass kein Modell platziert wird.

- Der Nutzer kann wieder in die QR Scanning Szene zurückspringen und es mit einem anderen Modell probieren. SONST KEIN FALLBACK

- im Repo unter den Artefakten zu finden

- technisch
Starten des Feuerwerks

- Das ParticleSystem des in der Szene platzierten Feuerwerks wird einmalig abgespielt.

- Das ParticleSystem wird abgespielt und wird automatisch beendet.

- Das Modell besitzt nicht die vorgeschriebene Struktur und das ParticleSystem wird nicht gestartet

- Der Nutzer versucht das Gleiche mit einem anderen Modell.
- (a) Der Nutzer bricht dem Vorgang ab, um das nächste Feuerwerk zu platzieren.

- technisch
"Firework-Visualisation and Sound"
- technisch, kommunikativ, kompetenzorientiert

- Die VFX-Ausführung wird akustisch begleitet.

- Neben den VFX wird zum richtigen Zeitpunkt SFX ausgeführt

- (a) SFX werden nicht oder zeitlich versetzt abgespielt.
- Das Feuerwerk wird korrekt für alle teilnehmenden Geräte angezeigt.
- Zuschauergeräte erhalten falsche oder keine Informationen.

- zur Atomarisierung des funktionalen Prototyps

- Überarbeitung der PoCs
 - neue Funktionen wurden hinzugefügt
 - nicht brauchbare oder realisierte Funktionen wurden gestrichen



Cognitive Walkthrough

Nutzertauglichkeit



Cognitive Walkthrough

1. Gelangen in die Kameraszene

Success story:

Durch den übersichtlichen Homescreen und der wenigen Buttons sollte es einfach sein auf den Knopf „Start Camera“ zu drücken.

2. Scannen eines QR Codes

Success story:

Aufgabe: Durch den klaren Ausdruck des Knopfes „Scan QR Code“ sollte der Nutzer wissen, dass dieser jetzt einen QR scannen muss.

Erkennung: Durch die integrierte Taschenlampe sollte der QR Code auch im Dunkeln scanbar sein.

Failure story:

Permission: Der Nutzer lehnt die Erlaubnis zur Nutzung der Kamera ab und muss möglicherweise die App neu starten, um sie diesmal zu bestätigen.

Erkennung: Der Nutzer hat den QR Code nicht vollständig vor der Kamera und ist sich dessen nicht bewusst.

Behebungsmöglichkeit: Eine Benachrichtigung für den Nutzer, dass der QR Code zu erkennen ist oder nicht.

- Mit dem Cognitive Walkthrough wollten wir die Aufgaben analysieren, die sich im Prototypen befinden und ob die Bewältigung dieser nutzertauglich sind
 - Aufgabe beschrieben
 - je nach dem jeweils Succes Storys und Failure Storys mit reingebracht
 - zu den Failure Storys haben wir Möglichkeiten zu Problembehebungen vorgeschlagen, die im Falle einer vollwertigen Version beachtet werden müssten



Cognitive Walkthrough

Nutzertauglichkeit

3. Gelangen zur AR Szene

Failure Story: Dem Nutzer ist nicht klar, was der Pfeil bedeutet.

Behebungsmöglichkeit: Eine Benachrichtigung für den Nutzer, dass er nun mit dem Pfeil zur AR Szene gelangt.

4. Platzieren des Feuerwerks

Success Story: Der QR Code für die festgelegte Position des Feuerwerks sollte für den Nutzer erkennbar sein. Dieser kann dann den Knopf „Place Firework“ bestätigen.

Failure Story:

QR Code: Der QR Code ist im Bild nicht zu sehen, worauf das Feuerwerk nicht platziert werden kann

Behebungsmöglichkeit: Eine Benachrichtigung für den Nutzer, dass der QR Code im Bild zu sehen sein soll.

Erkennung: Der QR Code ist für die Kamera nicht erkennbar genug, da es zu dunkel ist.

Behebungsmöglichkeit: Das Fertigstellen von Displays für den QR Code, die beleuchtet sind.

- Mit dem Cognitive Walkthrough wollten wir die Aufgaben analysieren, die sich im Prototypen befinden und ob die Bewältigung dieser nutzertauglich sind
 - Aufgabe beschrieben
 - je nach dem jeweils Succes Storys und Failure Storys mit reingebracht
 - zu den Failure Storys haben wir Möglichkeiten zu Problembehebungen vorgeschlagen, die im Falle einer vollwertigen Version beachtet werden müssten



Cognitive Walkthrough

Nutzertauglichkeit



5. Starten des Feuerwerks

Success Story: Durch den klaren Ausdruck des Knopfes „Start Firework“ sollte der Nutzer wissen, dass dieser jetzt das Feuerwerk starten kann.

Failure Story: Der Nutzer unbewusst betätigt mehrere Male den Knopf „Start Firework“, sodass das Feuerwerk abgebrochen wird.

Behebungsmöglichkeiten: Eine Benachrichtigung für den Nutzer, dass das Feuerwerk abgebrochen wurde.

6. Gelangen in den Home Screen

Success Story: Die Pfeile gegen die Leserichtung signalisieren dem Nutzer, dass dieser in den Home Screen gelangen kann.

- Mit dem Cognitive Walkthrough wollten wir die Aufgaben analysieren, die sich im Prototypen befinden und ob die Bewältigung dieser nutzertauglich sind
 - Aufgabe beschrieben
 - je nach dem jeweils Succes Storys und Failure Storys mit reingebracht
 - zu den Failure Storys haben wir Möglichkeiten zu Problembehebungen vorgeschlagen, die im Falle einer vollwertigen Version beachtet werden müssten



Vision-Exposé für einen Werbefilm

- Werbefilm-Exposé zum Bewerben des Endproduktes
- Charaktere aus den Personas
- Message:
 - Gemeinschaftsgefühl und Freude durch TSF
 - Aufzeigen der Funktionalität von TSF
 - Nicht schädlich für Menschen mit Hörschäden (Juliane)
 - Alter der Zielgruppe ist von jung bis alt

Exposé für einen Werbefilm

Für das Modul Entwicklungsprojekt

im WS 2021/2022

bei Prof. Hans Kornacher

Gruppenname: Tidy Shiny Fireworks

Gruppenteilnehmer: Kevin Repke, Paul André Johne

Motivation: Im Rahmen des Modul Entwicklungsprojekt (Wintersemester 2021/2022) bei Prof. Hans Kornacher soll ein Entwicklungsprojekt implementiert werden. Dazu soll es ein Konzept für die Bewerbung dieses geben in Form eines Werbefilms

Arbeitstitel: Tidy Shiny Fireworks - Vision

Genre: Produktbewerbung

Hauptcharaktere:

Fabio, Theresa, Juliane (Charakterbeschreibungen sind den Personas zu entnehmen, nur das der Wohnort Gummersbach ist)

Handlung:

Es ist Silvesterabend und die Menschen in Gummersbach bereiten sich auf den Abend vor. Fabio wird von seiner Mutter warm angezogen. Theresa macht sich fertig und geht zu ihren Mitbewohnern, die im Foyer stehen. Sie sind bereit, um rauszugehen. Juliane ist schon fertig, schaut in den Himmel aus dem Fenster und steckt sich ihr Hörgerät ins Ohr. Sie wird von ihren Kindern gerufen, um loszugehen. Fabio, Theresa und Juliane gehen mit ihren Begleitungen in die Innenstadt, wo viele weitere Bewohner aufzufinden sind. Es kommt eine Durchsage mit dem Countdown. Alle drei holen ihre Handys raus, starten die App „Tidy Shiny Fireworks“ und richten diese Richtung Himmel. Der Countdown ist abgelaufen und am Himmel ist nichts zu sehen. Fabio schaut auf sein Handy und sieht das virtuelle Feuerwerk. Alle springen auf, schauen sich auf den Handys das Feuerwerk an und stoßen miteinander an. Gelächter und Freude ist bei den Anwesenden zu sehen. Juliane ist erstaunt und hält sich am Hörgerät. Die virtuellen Explosionen sind nicht zu laut für sie. Sie freut sich und feiert weiter.

Ort, Datum: Hemer, 18.02.2022

Autoren: Kevin Repke, Paul André Johne

- weiteres zusätzliches Artefakt ist Exposee für einen Werbefilm für die Produktbewerbung einer vollwertigen Version
- Charaktere aus den Personas
- Story:
 - die Menschen bereiten sich auf Silvester vor
 - treffen sich in der Innenstadt
 - kurz vor Mitternacht starten alle die App
 - Obwohl am Himmel kein Feuerwerk zu sehen ist, entertaint die App die Leute mit dem virtuellen Feuerwerk
 - Freude und Feier
- Message:
 - Gemeinschaftsgefühl und Freude durch TSF
 - Aufzeigen der Funktionalität von TSF
 - nicht schädlich für Menschen mit Hörschäden (Juliane)
 - Alter der Zielgruppe ist von jung bis alt



Funktionaler Prototyp

Vorstellung

[https://www.youtube.com/
watch?v=We8S8Z0OOxI](https://www.youtube.com/watch?v=We8S8Z0OOxI)



Funktionaler Prototyp

Android SDK Wrapper



```
4 references
15 public class AndroidCameraLIB {
16
17     AndroidJavaClass unityActivity = new AndroidJavaClass("com.unity3d.player.UnityPlayer");
18
19     AndroidJavaClass toastClass = new AndroidJavaClass("android.widget.Toast");
20     AndroidJavaClass cameraClass = new AndroidJavaClass("android.hardware.Camera");
21
22     AndroidJavaObject toast;
23     string backCamId = "none";
24
25     4 references
26     public void makeToast(string message) {
27         /* Preparing parameters for toast.makeText */
28         object[] parameters = new object[3] {
29             unityActivity.GetStatic<AndroidJavaObject>("currentActivity"),
30             message,
31             toastClass.GetStatic<int>("LENGTH_SHORT")
32         };
33
34         /* Creating toast and showing it on current activity */
35         toast = toastClass.CallStatic<AndroidJavaObject>("makeText", parameters);
36         toast.Call("show");
37
38     1 reference
39     public void activateTorch() {
40         var context = unityActivity.GetStatic<AndroidJavaObject>("currentActivity");
41         var cameraManager = context.Call<AndroidJavaObject>("getSystemService", "camera");
42
43         try {
44             var camIDs = cameraManager.Call<string[]>("getCameraIdList");
45             backCamId = camIDs[0];
46         } catch {
47             this.makeText("Getting Back Camera ID failed");
48         }
49
50         if (backCamId != "none") {
51             cameraManager.Call("setTorchMode", new object[] { backCamId, true });
52         }
53     }
54 }
```

- AndroidCameraLIB stellt für das weitere Projekt Funktionen bereit, um den AndroidJava-Spaghetti Code an einer Stelle gesammelt zu halten
- PROBLEM:
auf Kamera kann nur 1x zugegriffen werden
=> Problem mit AR und WebCamTexture

- Einbindung der AndoidCameraLibrary für Funktionen bezüglich AndroidJava
 - makeToast() zur Erschaffung von einfachen Notifications
 - activateTorch() für die Aktivierung der Lampe
- Auf Kamera kann nur einmal zugegriffen werden
 - Konflikt mit AR un WebCamTexture



Funktionaler Prototyp

Levelloader

```
private void LoadNewScene(string nextScene) {
    StartCoroutine(LoadingScene(nextScene));
}

1 reference
private void LoadNewScene(int nextSceneIndex) {
    StartCoroutine(LoadingScene(nextSceneIndex));
}

2 references
IEnumerator LoadingScene<T>(T scene) {
    AsyncOperation asyncLoadScene;

    transition.SetTrigger("sceneLoading");

    if (scene is string)
        asyncLoadScene = SceneManager.LoadSceneAsync(scene as string);
    else if (scene is int) {
        int index = (int)(object)scene;
        asyncLoadScene = SceneManager.LoadSceneAsync(Convert.ToInt32(scene));
    } else {
        throw new Exception("LoadingScene<T>() only accepts string or int");
    }

    /* Callback when scene was loaded successfully */
    asyncLoadScene.completed += (AsyncOperation it) => {
        Debug.Log("Fertig");
        transition.SetTrigger("sceneLoaded");
    };

    asyncLoadScene.allowSceneActivation = false;

    /* Wait here while animation hasn't finished yet */
    while (!(transition.GetCurrentAnimatorStateInfo(0).IsName("WipeFade_Start") && transition.GetCurrentAnimatorStateInfo(0).normalizedTime > 1 - 1e-6)) {
        yield return null;
    }

    asyncLoadScene.allowSceneActivation = true;

    while (!(transition.GetCurrentAnimatorStateInfo(0).IsName("WipeFade_End") && transition.GetCurrentAnimatorStateInfo(0).normalizedTime > 1 - 1e-6)) {
        yield return null;
    }

    Destroy(gameObject);
}
```

- generisch, um Laden über Namen oder Index zu ermöglichen
- 2. Teil der Animation in neuer Szene abgespielt

- Levelloader nötig, um das Wechseln zwischen den Scenes zu managen
- schwarzer Kreis (Gameobject) als Animation zwischen den Szenen



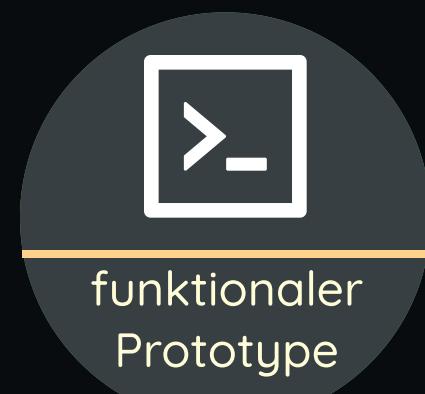
Funktionaler Prototyp

QR Reader 1/2



```
private void Start() {
    /* Code for Android Version */
#if PLATFORM_ANDROID
    if (!Permission.HasUserAuthorizedPermission(Permission.Camera))
        Permission.RequestUserPermission(Permission.Camera);

    camFunctions = new AndroidCameraLIB();
#endif
    qrReader = new BarcodeReader();
}
```



```
private void UpdateCameraRender() {
    if (!_isCamAvailable) {
        if (Permission.HasUserAuthorizedPermission(Permission.Camera))
            SetCamera();
        return;
    }

    /* Updates video if flipped */
    // Set aspectRatio
    float ratio = (float)_camTexture.width / (float)_camTexture.height;
    aspectRatioFitter.aspectRatio = ratio;
    // Rotation of raw image
    int orientation = -_camTexture.videoRotationAngle;
    // Z-Roll
    camImage.rectTransform.localEulerAngles = new Vector3(0, 0, orientation);
}
```



```
private void SetCamera() {
    // Trying to access cameras
    WebCamDevice[] devices = WebCamTexture.devices;

    if (devices.Length == 0) {
        _isCamAvailable = false;
        return;
    }

    for (int i = 0; i < devices.Length; i++) {
        if (!devices[i].isFrontFacing) {
            _camTexture = new WebCamTexture(devices[i].name,
                (int)scanArea.rect.width,
                (int)scanArea.rect.height);
            break;
        }
    }

    // Starting back camera
    if (_camTexture != null) {
        _isCamAvailable = true;
        _camTexture.Play();
        camImage.texture = _camTexture;
        camImage.color = new Color(255, 255, 255, 255);
    }
}
```

- BarcodeReader von ZXing genutzt, um Bildmaterial zu analysieren
- Bildmaterial über WebCamTexture von Unity erhalten

- bei Start()
 - Erlaubnis nach Kamerazugriff in der App
 - Instantiierung des BarcodeReaders
- Rendern des Kamerabildes nach Ratio (falls App im horizontalen Modus sich befindet)
- bei SetCamera()
 - Zugriff auf Kamera und dessen Start
 - Übergabe des Kamerabildes mittels WebCamTexture



Funktionaler Prototyp

QR Reader 2/2



```
0 references
public void OnClickScan() {
    threadOfScan = new Thread(Scan);
    threadOfScan.Start();
}

0 references
public void OnClickStopScan() {
    threadOfScan.Abort();
    textDebug.text = "Stopped scanning";
    buttonStopScan.SetActive(false);
    buttonStartScan.SetActive(true);
}
```

- Scanning auf separaten Thread, da sonst die Darstellung des Bildmaterials an Performance verliert

```
private void Scan() {
    Result qrContent;
    textDebug.text = "Scanning..";
    buttonStartScan.SetActive(false);
    buttonStopScan.SetActive(true);
    buttonStartDownload.SetActive(false);

    do {
        qrContent = qrReader.Decode(_camTexture.GetPixels32(), _camTexture.width, _camTexture.height);
        if (qrContent != null)
            textDebug.text = qrContent.Text;
    } while (qrContent == null);

    buttonStopScan.SetActive(false);
    buttonStartScan.SetActive(true);
    buttonStartDownload.SetActive(true);

    if (File.Exists(Path.Combine(Application.persistentDataPath, textDebug.text))) {
        Downloader.filePath = Path.Combine(Downloader.storagePath, textDebug.text);
        buttonToAR.SetActive(true);
    }
}
```

- BarcodeReader.Decode() nutzt das Bildmaterial als 32 Bit RGB Array

- bei Scan()
 - zusätzlich noch Überprüfung, ob Modell in der App schon existiert



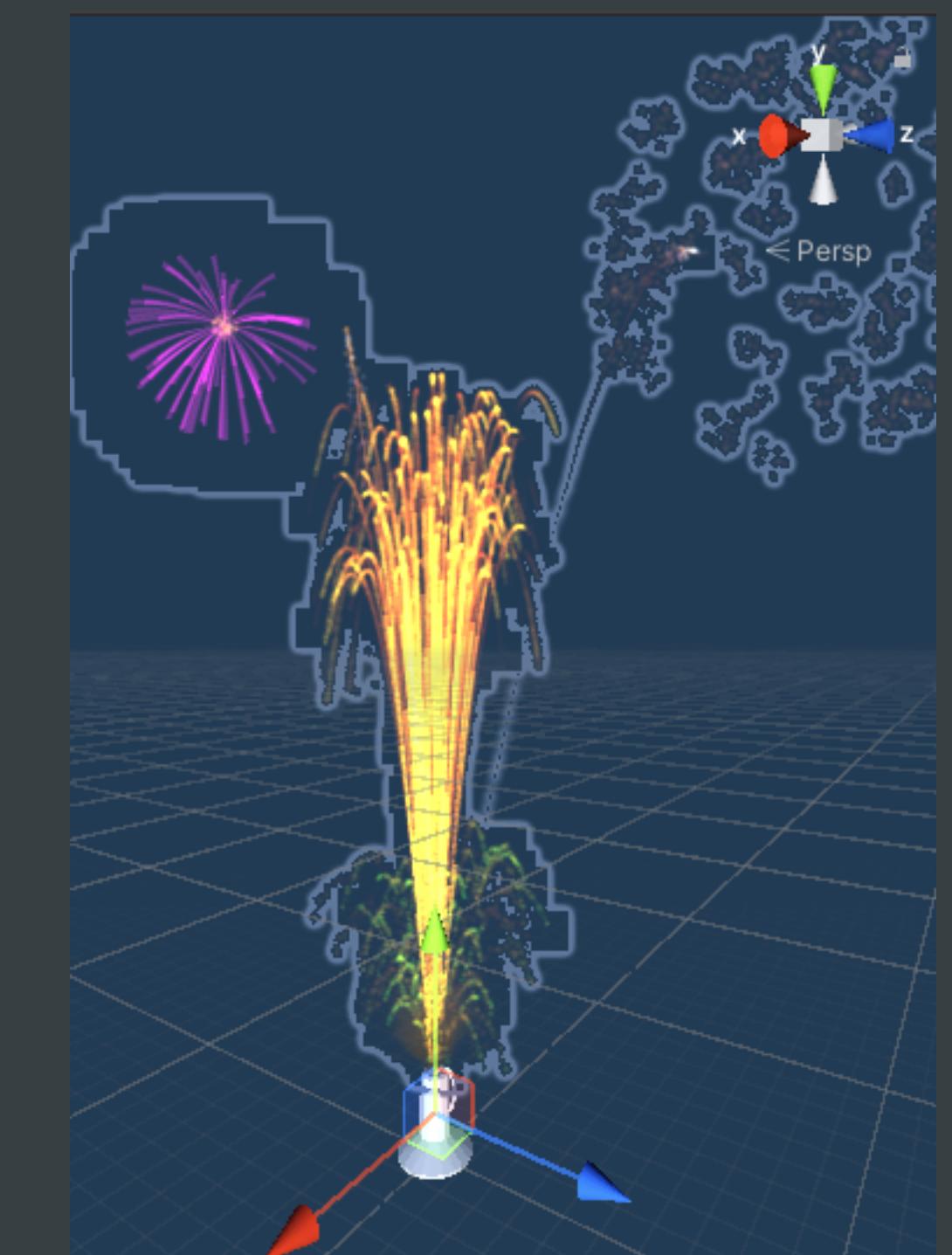
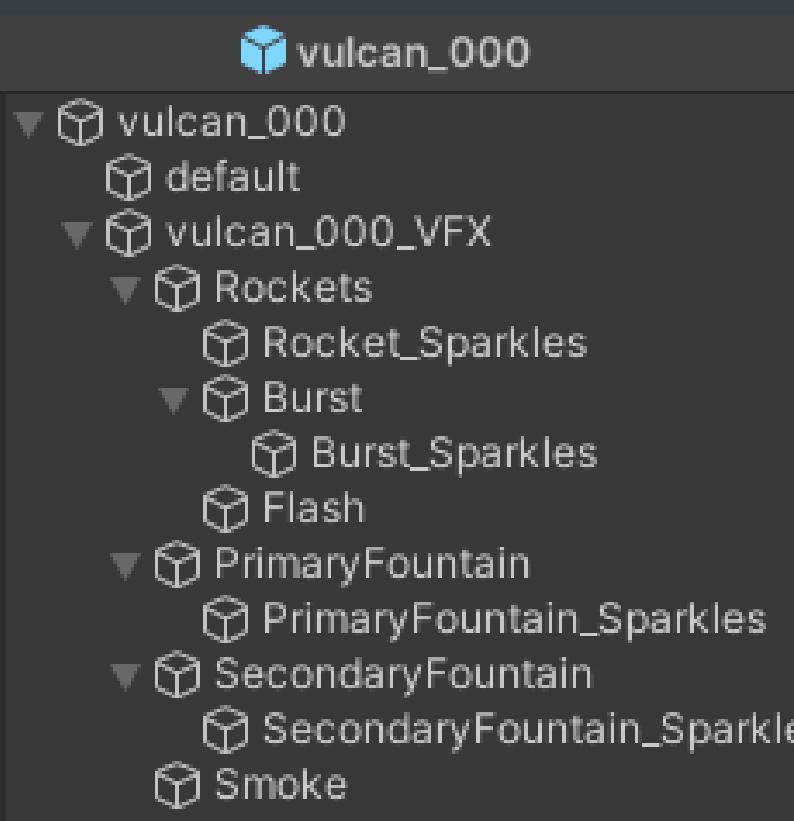
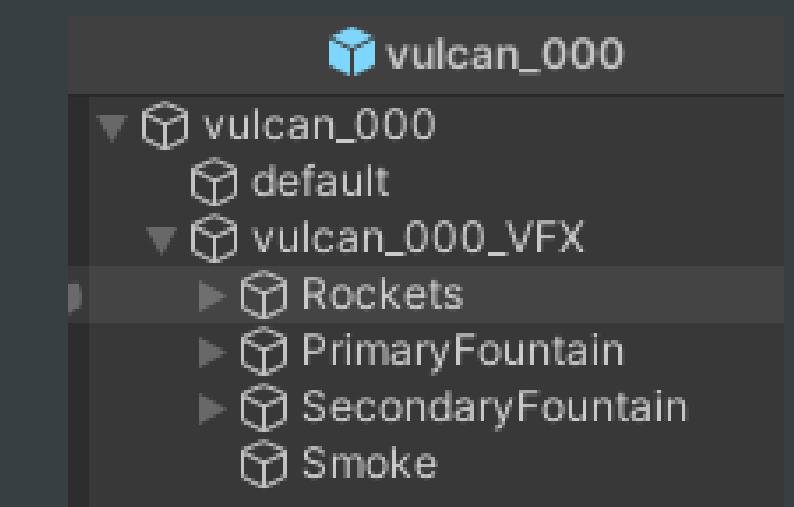
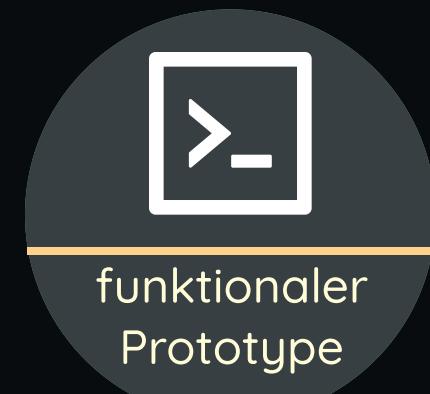
Funktionaler Prototyp

Particle Systems

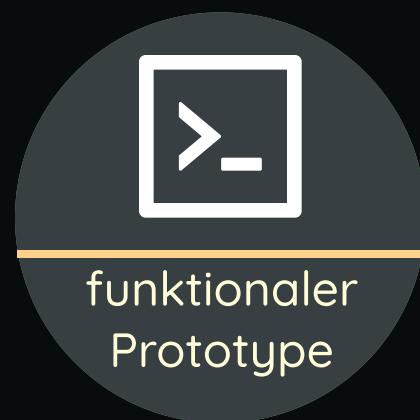
- Feuerwerk aus Modell (z.B. Batterie) und VFX
- Darstellung der VFX mittels ~~VFX Graph~~ Particle System
- Ergebnisse auf Smartphones sichtbar, da PS auf die CPU zugreift



- Aufbau von vulcan_000
 - 3D Model
 - Raketen
 - 2 Fontänen
 - Rauch
- Versch. Material für Partikel



- Feuerwerk = Modell + Particle Systems
- Statt VFX Graph, dessen Effekte auf die GPU greifen, nutzen wir Particle System, dessen Effekte auf die CPU greifen
 - ein Nachteil: begrenzte Anzahl an Partikeln im Vergleich zum VFX Graph
 - ein Vorteil: im Editor gegeben, kein externes Package nötig
- Aufbau von vulcan_000 angepasst
 - mehr Effekte
- die unterschiedlichen PS benötigen jeweils bestimmte Materialien
 - meisten Hauptmaterial für Partikel + zusätzliches Material für Pfade



Funktionaler Prototyp

Exkurs: Asset Bundles

- Datei, die non-code Assets (Modelle, Prefabs, Texturen etc.) archiviert
- Bundles sind platform-spezifisch
- Zur Erstellung dieser ist im Editor eine Funktion nötig, die alle gelabelten Prefabs zu Asset Bundles komprimiert
(CreateAssetBundles)
- Speicherung dieser in Servern zur Effizienzsteigerung (z.B. Anpassung ohne App-Updates)

```
public class CreateAssetBundles
{
    public static string assetBundleDirectory = "Assets/AssetBundles/";

    [MenuItem("Assets/Build AssetBundles")]
    0 Verweise
    static void BuildAllAssetBundles()
    {
        if (Directory.Exists(assetBundleDirectory))
        {
            Directory.Delete(assetBundleDirectory, true);
        }

        Directory.CreateDirectory(assetBundleDirectory);

        BuildPipeline.BuildAssetBundles(assetBundleDirectory, BuildAssetBundleOptions.None, BuildTarget.Android);
        AppendPlatformToFileName("Android");
        Debug.Log("Android bundle created...");

        RemoveSpacesInFileNames();

        AssetDatabase.Refresh();
        Debug.Log("Process complete!");
    }

    1 Verweis
    static void RemoveSpacesInFileNames()...

    1 Verweis
    static void AppendPlatformToFileName(string platform)...
}
```

- Arbeit mit AssetBundles ermöglicht es, die Modelle außerhalb des Projektes zu lagern und zu bearbeiten
- Datei für Archivierung von non-code Assets wie Modelle, Prefabs, Texturen, Bilder, etc.
- Hinzufügen der Funktion CreateAssetBundles im Editor, um Komprimierung zu ermöglichen
- Effizientsteigerung, da
 - keine App-Updates nötig bei kleinen Anpassungen
 - Lagerung nicht im Projekt => Größe der App geringer
 - Über Runtime auf Abruf bereit



Funktionaler Prototyp

weiter eingegangen: Downloader

```
private void Start() {
    storagePath = Path.Combine(Application.persistentDataPath, "3DModels");

    if (!Directory.Exists(storagePath)) {
        Directory.CreateDirectory(storagePath);
        textDebug.text = "Directory for fireworks was created";
    } else {
        textDebug.text = "Directory for fireworks exists on your device";
    }
}

0 references
public async void OnClickDownload() {
    string scannedID = textDebug.text;

    if (!File.Exists(Path.Combine(storagePath, scannedID))) {
        buttonStartDownload.SetActive(false);
        await DownloadAssetBundle(scannedID);
    } else {
        filePath = Path.Combine(storagePath, scannedID);
        textDebug.text = "The scanned Firework already exists on your device";
        buttonStartDownload.SetActive(false);
    }

    buttonToAR.SetActive(true);
}

1 reference
private async Task DownloadAssetBundle(string currentID) {
    var downloadLink = $"{rawDownloadLink}{currentID}";
    var taskInspector = new Task<UnityWebRequest>[1];

    UnityWebRequest request = UnityWebRequest.Get(downloadLink);

    taskInspector[0] = StartDownload(request, currentID);
    await Task.WhenAll(taskInspector);

    if (taskInspector[0].Result.result != UnityWebRequest.Result.Success)
        textDebug.text = $"Download failed: {request.error}";
    else
        textDebug.text = "Finished download";
}

1 reference
private async Task<UnityWebRequest> StartDownload(UnityWebRequest request, string fileName) {
    filePath = Path.Combine(storagePath, fileName);
    request.downloadHandler = new DownloadHandlerBuffer();

    var operation = request.SendWebRequest();
    while (!operation.isDone)
        await Task.Yield();

    if (request.result == UnityWebRequest.Result.Success) {
        File.WriteAllBytes(filePath, request.downloadHandler.data);
    }
    return request;
}
```

- UnityWebRequest besitzt Handler, der erhaltene Daten beinhaltet
=> File.WriteAllBytes() speichert Handlerdaten persistent ab
- Daten im automatisch angelegten Ordner abgelegt

- Bei uns sind die Modelle auf Google Drive gelagert
 - haben individuelle Google Drive ID, womit dann Downlaod möglich ist



Funktionaler Prototyp

ARLogic

```
public void OnPlaceObject() {
    LoadPrefabFromAsset();

    trackedImageManager = sessionOrigin.gameObject.AddComponent<ARTrackedImageManager>();
    trackedImageManager.referenceLibrary = trackedImageManager.CreateRuntimeLibrary(xrRefImgLib);
    trackedImageManager.requestedMaxNumberOfMovingImages = 1;
    trackedImageManager.trackedImagePrefab = trackedImagePrefab;
    trackedImageManager.enabled = true;

    buttonPlaceFirework.SetActive(false);
    buttonIgniteFirework.SetActive(true);
}

1 reference
private void LoadPrefabFromAsset() {
    // buttonPlaceFirework.SetActive(false);
    AssetBundle loadedAssetBundle = AssetBundle.LoadFromFile(currentFilePath);

    if (loadedAssetBundle == null) {
        textdebug.text = "Failed to load AssetBundle!";
        return;
    }

    textdebug.text = "AssetBundle was loaded";

    trackedImagePrefab = loadedAssetBundle.LoadAsset<GameObject>(loadedAssetBundle.GetAllAssetNames().GetValue(0) as string);
    loadedAssetBundle.Unload(false);

    textdebug.text = $"Firework placed on QR Code: {trackedImagePrefab.transform.hierarchyCount}";
}

0 references
public void OnIgniteObject() {
    foreach (var i in trackedImageManager.trackables) {
        i.gameObject.transform.GetChild(1).GetComponent<ParticleSystem>().Play();
    }
    textdebug.text = "BOOM!";

    buttonIgniteFirework.SetActive(false);
}
```

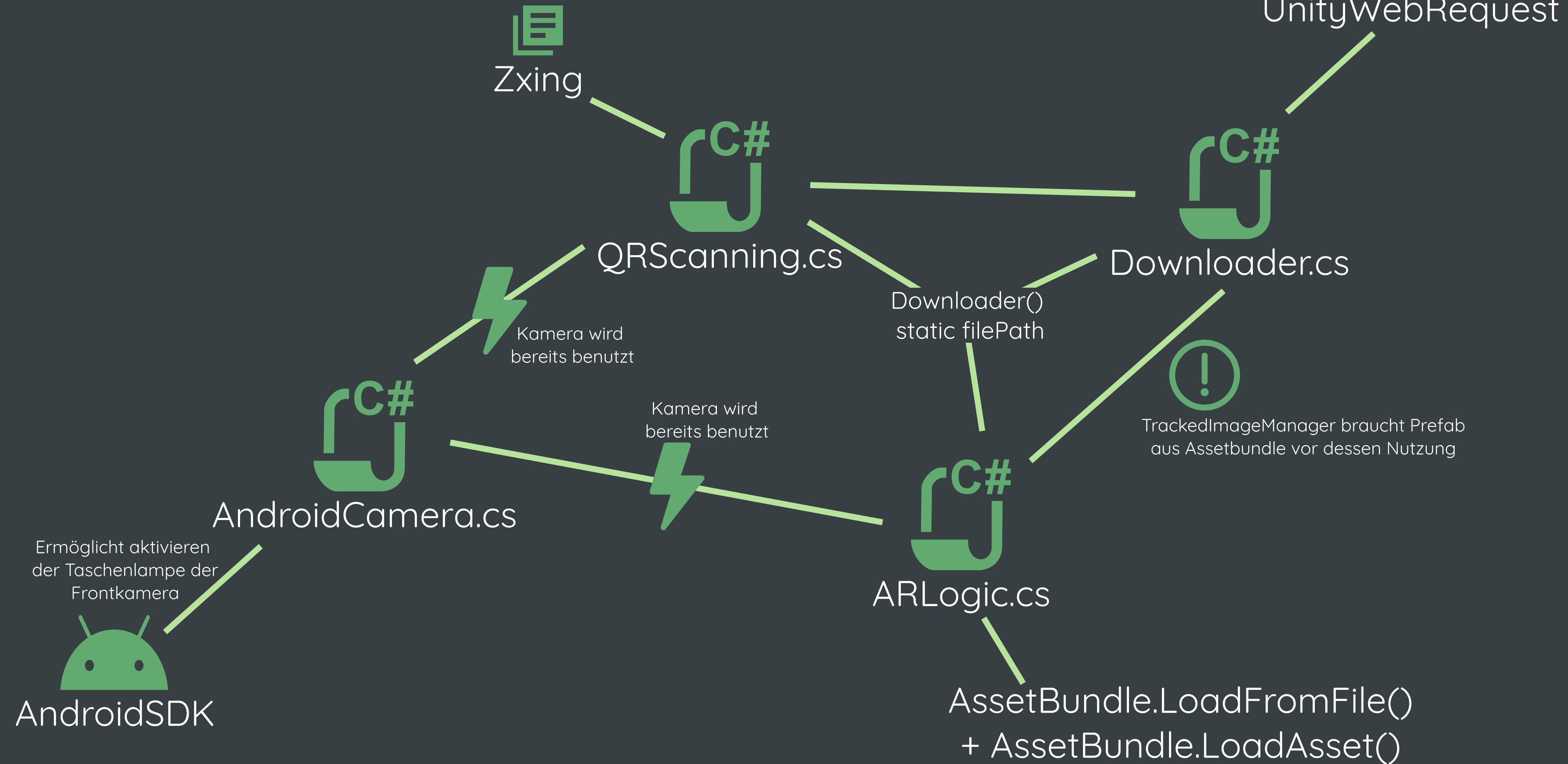
- AR Origin erhält zur Laufzeit Manager für TrackedImages als Komponente
- aus Prefab wird Gameobject vom Manager erstellt

- bei LoadPrefabFromAsset()
 - Laden des gedownloadeften AssetBundles mit LoadFromFile()
 - Laden des Feuerwerkmodells mit LoadAsset()
 - Unload(false) setzt AssetBundle frei ohne die zuvor geladenen Assets zu löschen
- bei OnIgniteObject()
 - Zugriff auf die Particle System Childs im Feuerwerksmodell, um VFX zu starten



Überblick der Konflikte

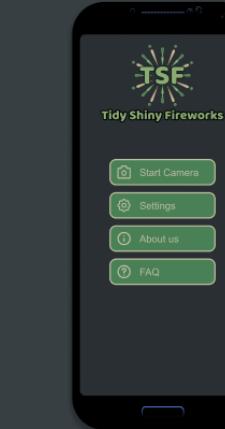
Problem bei der Realisierung





Plakat Übersicht

- Beinhaltet die wichtigsten Informationen
 - Problemdomäne
 - Lösungsidee
 - Architektur
 - Ziele
 - User-Journey
- Gestaltet nach unserem Styleguide



1. Home

Im Home Bildschirm hat der Nutzer die Möglichkeit, zur Kamerazene, zu den Optionen, zu den Informationen über die App oder zum FAQ-Bereich zugelangen.



2. Scan QR Code

Der Nutzer kann mit dem QR Scanner einen QR Code scannen, welcher in der Veranstaltung ausgestellt ist und die für diesen angepasstes und herunterladbares Feuerwerk beinhaltet.



3. Place Firework

Nach dem Herunterladen wird der QR Code als Positionspunkt für das Feuerwerk benutzt. Der Nutzer kann auf Knopfdruck diesen platzieren.



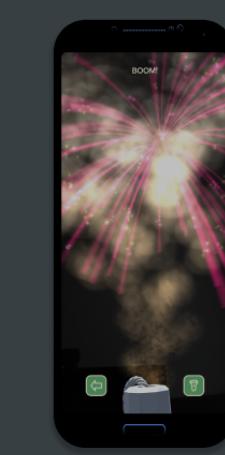
4. Firework is placed

Das Feuerwerk wurde nach Betätigen des Knopfes platziert.



5. Start Firework

Der Nutzer kann nun das Feuerwerk starten.



6. Appreciate Firework

Jetzt kann der Nutzer das Feuerwerk genießen!

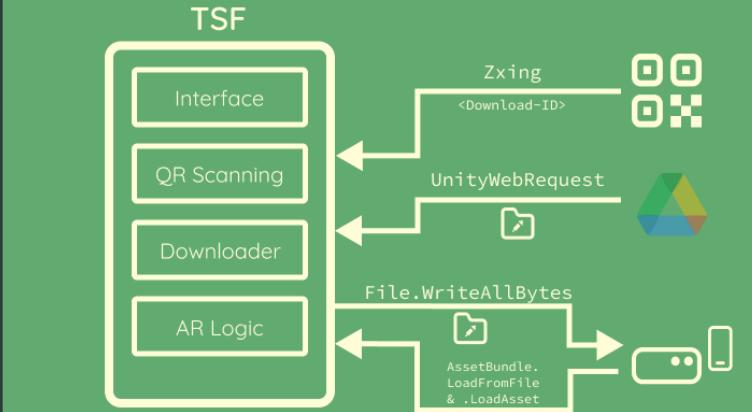
Die Problemdomäne:

Die Belastung des Klimas nimmt in der heutigen Zeit mehr an Bedeutung zu. Neue Maßnahmen müssen entworfen und Technologien umgesetzt werden, um die Zerstörung des Ökosystems zu verhindern bzw. zu minimieren. Vor allem in Bezug auf das Silvester-feuerwerk 2019/2020 sind diese umgesetzt worden wie z.B. in Deutschland ein Verkaufsverbot von Feuerwerk und in Amsterdam das private Raketen- und Böllerverbot. Auch wenn diese eigentlich im Hinblick der COVID-19 Pandemie beschlossen wurden, um die Krankenhäuser zu erleichtern, sind positive Effekte wahrzunehmen wie die Einsparung bei der Feinstaubbelastung oder der Verringerung des Müll-aufkommens um ca. 3500 Tonnen in Deutschland. Nun, da sich die Pandemie dem Ende nähert, treten Überlegungen voran, ob gewisse Maßnahmen in den nächsten Jahren weiter fortgesetzt oder alternative Technologien erbracht werden sollen, um die Vorteile weiter ausschöpfen zu können oder auszubauen.

Unsere Lösung:

Mit der App „Tidy Shiny Fireworks“ soll es mit Augmented Reality möglich sein, Feuerwerk auf dem Smartphone anzeigen zu lassen. Dafür werden Feuerwerksmodelle über QR Codes bereitgestellt, die der Nutzer über den integrierten QR Scanner unterladen kann. In der Szene wird dann der besagte QR Code auch als Referenzpunkt für das Modell benutzt, welches dann daraufhin gestartet werden kann. Die App kann bei öffentlichen oder privaten Veranstaltung genutzt werden, z.B. im Stadtzentrum am Silvesterabend im Form einer Stadtfest, die von der Stadt organisiert wird.

Die Architektur:



Die App lässt sich in 4 Bestandteile aufteilen. Das „Interface“ kümmert sich um die Navigation in den Screens. Das „QR Scanning“ benutzt **Zxing**, um auf den Inhalt des QR Codes zuzugreifen, welcher für das Downloaden des AssetBundles mit dem Feuerwerk mittels **UnityWebRequest** nötig ist. Die „AR Logic“ sorgt für das Abspeichern des Bundles und kann jederzeit auf dessen Inhalt zugreifen, um schlussendlich das Feuerwerk in der Szene zu platzieren.

Unser Ziel:

Die Bereitstellung der App für Veranstaltungen jeglicher Art soll einen Ausgleich zu den Restriktionen von Feuerwerk besonders für urbane Regionen schaffen. Eine umweltfreundliche und der Gesundheit nicht schädende Alternative solcher Art soll es ermöglichen, weiterhin die „Feuerwerkstradition“ in abgewandelter Form genießen zu können, ohne dabei Feinstaubpartikel in die Luft zu schießen, die Lautstärke begrenzt zu halten und keinen Müll zu produzieren.

- Jetzt schon vorstellen, da schon fertiggestellt
- beinhaltet die wichtigsten Informationen
 - Problemdomäne, mit welchem Problem sich TSF befasst
 - Lösungsidee, wie TSF konzipiert ist
 - eine Skizze zur Architektur, wie TSF läuft
 - Ziele, die wir uns gesetzt haben
 - User-Journey, wie ein üblicher Ablauf mit TSF aussieht
- Gestaltet nach unserem Styleguide
 - da schon seit Beginn des Projektes von uns erstellt worden
 - Individualität gegenüber anderen Plakaten



Erweiterungsmöglichkeiten

für eine vollwertige Version

- Die anderen Buttons im Homescreen mit Funktionalität füllen
- In der QR Scanning Szene einen Menübutton, um bereits vorhandene Feuerwerksmodelle auszuwählen für die AR Szene
- Die UI in der AR Szene reparieren, sodass das Feuerwerk hinter dem Button ist
- Interne Erstellung von QR Texturen mit Zxing, um diese als Referenz beim Image Tracking zu nutzen
- Hinzufügen von Audio (SFX)



Fazit und kritische Reflexion zum Projekt

- Sehr zufrieden mit dem Projektergebnis
- Sehr coole Idee, die möglicherweise wirklich so in der Gesellschaft genutzt werden könnte



ABER: **Nie wieder Augmented Reality!**



- Sehr aufwendige Recherche in der Entwicklung nötig
- Umsetzung besitzt hohe Komplexität
- Zusammenarbeit war ab und zu hakelig wegen Building Problemen und Fernstudium





Einleitung



Zusätzliche
Artefakte



finaler
Prototype



Finish

Vielen Dank für ihre Aufmerksamkeit!

