VRProject-alexh **Dokumentation**

26. Februar 2020

Zuallererst wurde auf GitHub ein Repository erstellt. Dabei wurde darauf geachtet, dass die „.gitignore“ Datei so aufgesetzt wird, dass jegliche Unity-Projekt-Dateien und -Ordner die nicht notwendig sind, sowie diverse Windows und macOS Systemdateien, nicht mit dem Online-Repository synchronisiert werden. Außerdem wurde die „readme.md“, dem Muster der vorherigen Projekte entsprechend, eingerichtet. **– Zeitaufwand: 1h**

Danach begann die Konzeptentwicklung für das Puzzle-Spiel. Meine erste Idee war es eine Art Quiz zu gestalten, bei dem die Spieler*innen die geographischen Lebensräume - in Form von Kontinenten – jeweils einer dazugehörigen Tierart zuordnen sollten. Am Beginn war es der Plan, dass die Tiere als 3D-Modelle in einem Raum stehen. Da ich allerdings auch etwas exotischere Tiere verwenden wollte, um das Puzzle nicht zu einfach zu gestalten, war es mir nicht möglich zu jeder Tierart ein geeignetes 3D-Modell zu finden. Da die Erstellung der Modelle aus eigener Hand nicht im zeitlichen Rahmen dieses Projekts lag, wurde die Entscheidung getroffen keine 3D-Modelle, sondern Bilder von den Tieren in einem Bilderrahmen an die Wände des Raumes zu platzieren. Um das Puzzle noch etwas aufzuwerten, kamen auch noch Namensschilder zur Anwendung. Diese müssen ebenso wie die Kontinente der entsprechenden Tierart zugeordnet werden. Die Kontinente können von einem Globus in Form von Platten entnommen werden. Die Namenschilder sind auf einem Tisch unter dem Globus platziert.
Am Ende wurden erste Entwicklungsinfos und die Kurzbeschreibung des Spiels in das Readme eingetragen. **– Zeitaufwand: 2h**

04. März 2020

Nun wurde recherchiert, welche Tierarten von den folgenden 7 Kontinenten ausgesucht werden: Nordamerika, Südamerika, Europa, Afrika, Asien, Australien, Antarktis. Dabei fiel die Wahl auf folgende Tierarten, respektive: Bison, Jaguar, Rentier, Nashorn, Roter Panda, Wallaby, Zügelpinguin. Es war nicht einfach die Tiere auszusuchen, da die Lebensräume von den meisten Tieren mittlerweile über mehr als einen Kontinent verteilt sind. In Anbetracht der natürlichen, ursprünglichen Herkunft der oben genannten Arten, ist eben diese Wahl entstanden.
– Zeitaufwand: 1,5h

Der nächste Schritt war es den Raum, in dem sich die Spieler*innen letztendlich befinden, mit Autodesk Maya zu modellieren. Dabei wurde in einem 10m*10m*3m großem Raum ein Tisch mit den Namensschildern und dem Globus in die Mitte positioniert. An den Wänden sind die 7 Bilder mit einem Sockel darunter symmetrisch aufgeteilt platziert. **– Zeitaufwand: 1.5h**

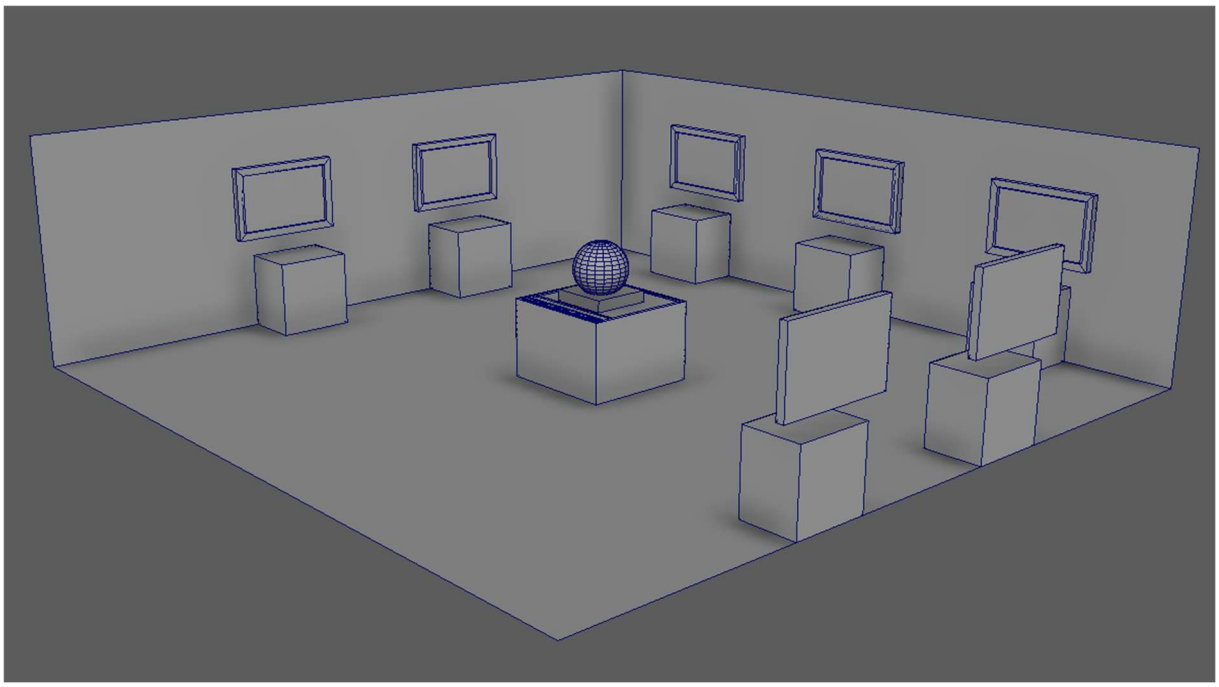


Abb. 1: Greybox-Model des Raumes in Autodesk Maya

11. März 2020

Das 3D-Modell wurde fertiggestellt und anschließend in die Unity-Szene importiert. Ein paar der Elemente wurden auch gleich mit einem Material ausgestattet. Als nächstes wurde das SteamVR Unity Plugin aus dem Asset Store importiert.
– Zeitaufwand: 1,5h

Bevor mit der Programmierung begonnen wurde, habe ich nach passenden Fotos der 7 Tiere gesucht. Anschließend wurden die Bilder dem Seitenverhältnis der Bilderrahmen (3:2) entsprechend, in Adobe Photoshop zugeschnitten.
– Zeitaufwand: 1,5h

18. März 2020

An diesem Tag wurde entschieden, dass aufgrund des Fernunterrichts, das Puzzle lediglich im 2D-Debug Modus von SteamVR komplett spielbar sein sollte. Es wurde also zuerst überprüft ob jegliche Gameplay-Mechaniken in diesem Modus umsetzbar sind. In meinem Fall musste ich keine erwähnenswerten Anpassungen vornehmen.
Das Readme wurde aktualisiert mit Screenshots, Informationen zur Zielplattform, zum benötigten Setup und zu Third Party Inhalten (Tierbilder und SteamVR Plugin). **– Zeitaufwand: 1h**

Danach wurden in Unity Materialen für die 7 Tierbilder erstellt, bei denen die Bilder als Albedo-Map eingefügt wurden. Die Materialen wurde den entsprechenden Leinwänden zugewiesen. Um die Bilder explizit zu beleuchten habe ich erste Spot-Lights in der Unity-Szene platziert. **– Zeitaufwand: 1,5h**
Zu diesem Zeitpunkt sah der Raum so aus:

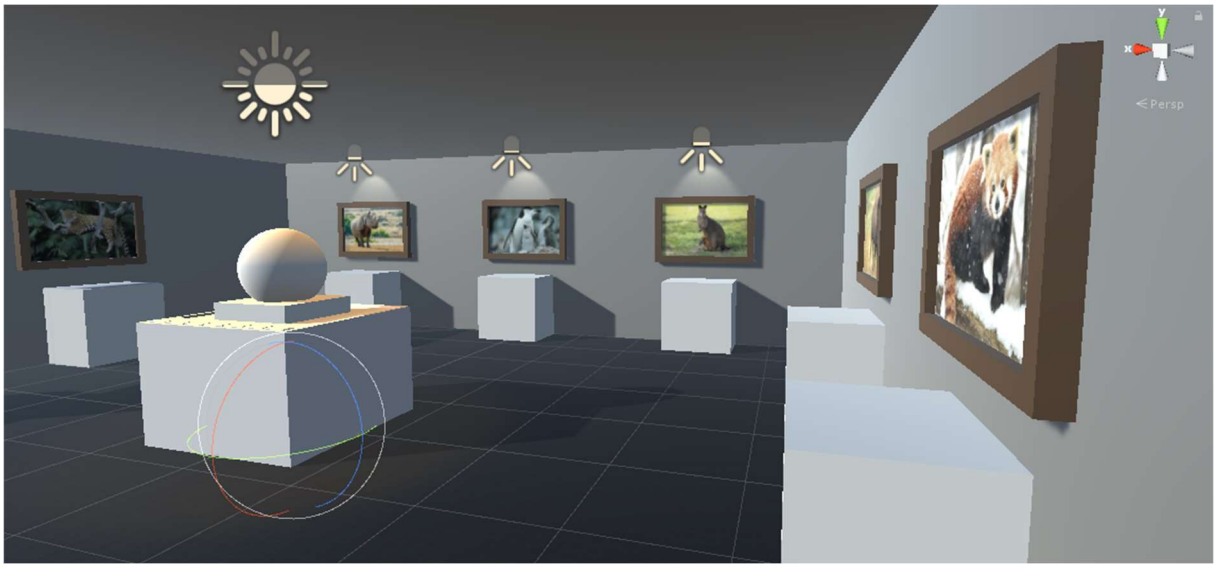


Abb. 2: Raum in Unity importiert und mit Tierbildern an den Wänden

Als nächstes wurde das „Player“ SteamVR Prefab eingebunden und die SteamVR Inputs wurden generiert. In Folge wurde der 2D-Debug Mode getestet. Dieser verhielt sich nicht wie erwartet: Im Play-Mode wurde dieser nicht automatisch aktiviert, das GameObject „FallbackObjects“ im Player Prefab musste manuell aktiviert werden, damit die Kamera des Debug-Modus verwendet wird. Außerdem konnte ich mit keinen Objekten interagieren, obwohl diese einen „Throwable“ und „Interactable“ Komponenten aufwiesen. **– Zeitaufwand: 0,5h**

25. März 2020

Mein Ziel war es nun das Problem mit dem Debug-Modus zu beheben. Zuerst versuchte ich mit einer Google-Suche Informationen zu diesem Problem zu finden. Überraschenderweise habe ich generell nur sehr wenig Einträge zum SteamVR Debug-Modus in Kombination mit der Unity-Engine gefunden. Ich versuchte also selbst das Problem zu lösen. Das erste was ich versuchte war auch bereits erfolgreich. Das SteamVR Plugin wurde vom Unity-Projekt entfernt und anschließend neu aus dem Asset Store importiert. Danach funktionierte alles einwandfrei: Der Debug-Modus aktivierte sich automatisch und die Interaktion funktionierte dieses Mal ebenfalls. **– Zeitaufwand: 1h**
Ich vermute, dass das Problem darin lag, dass das SteamVR Plugin ursprünglich auf einem anderen PC importiert wurde und durch die Übertragung des gesamten Projekts (inkl. des Plugins) über GitHub zu einem anderen PC ein Fehler auftrat.

Nun wurden diverse Box- und Mesh-Collider bei einigen Objekten, wie dem Boden, den Wänden, den Sockeln sowie dem Tisch und dem darauf liegenden Namensschildern hinzugefügt. **– Zeitaufwand: 1,5h**

Anschließend wurden den Namensschildern eine „Rigidbody“ und eine „Throwable“ Komponente hinzugefügt. Diese konnten nun also aufgehoben werden und auf den Sockeln unter den Bildern hingelegt werden. **Zeitaufwand: 0,5h**

13. April 2020

Zu Beginn wurden allen Elementen passende Farben bzw. Materialen zugewiesen.
- Zeitaufwand 0,25h

Als nächstes wurden den Namensschildern ein World Space Canvas angehängt, damit die Namen der Tiere auf den jeweiligen Objekten geschrieben stehen. Außerdem habe ich an der lehren Wand, hinter dem Spieler einen Text platziert, der die nötigen Anweisungen enthält. Über dem Globus in der Mitte des Raumes wurde ein „You Won!“ Text hinzugefügt, der sich einblendet, sobald das Puzzle erfolgreich gelöst wurde. Jegliche Textobjekte wurden mit dem TextMeshPro Plugin erstellt, da diese im Vergleich zu Standard-Unity UI Elementen anpassbarer sind und nicht das Problem haben, dass die Texte oft unscharf bzw. verpixelt dargestellt werden.
- Zeitaufwand: 1h

Der nächste Schritt war es jegliche Lichter zu platzieren und einzustellen. Das Hauptlicht ist ein Point-Light, welches den gesamten Raum abdeckt. Es ist in der Mitte des Raums platziert. Zusätzlich kommt von oben herab ein Spot-Light welches den Globus und den Tisch stärker beleuchtet, um den Fokus daraufzulegen.
Bei den 7 Bildern ist jeweils ein Spot-Light und ein Point-Light platziert. Diese sorgen dafür, dass die Bilder sowie die Sockel darunter ausreichend beleuchtet werden. Alle Objekte, die sich während der Spielphase in keiner Weise bewegen wurden „Static“ gesetzt und abschließend wurde eine Light-Map generiert.
– Zeitaufwand: 1h

Bevor die Programmierung gestartet wurde, musste noch eine letzte Sache erstellt werden: Der Globus mit separaten Kontinenten. Zuerst hatte ich vor ein bereits vorhandenes Model zu verwenden. Einen kostenlosen Globus mit separaten Kontinenten fand ich jedoch nicht. Ich musste diesen also selbst erstellen. Ich begann mit einer Kugel in Maya. Auf diese legte ich eine Globus Textur, um die Kontinent-Umrisse sehen zu können. Mit dem Multi-Cut Tool fügte ich nun an den Kontinent-Grenzen Edges hinzu. Die entstehenden Flächen die einen Kontinent darstellten, wurden extrahiert und in Folge extrudiert. Dadurch sind separate Kontinent-Platten entstanden. Zu guter Letzt wurden durch eine „Booleans > Difference“ Operation die Kontinent-Formen aus dem Globus herausgeschnitten, wodurch nach dem Entnehmen eines Kontinents eine Einkerbung in den Globus zu sehen ist. **– Zeitaufwand: 2,25h**

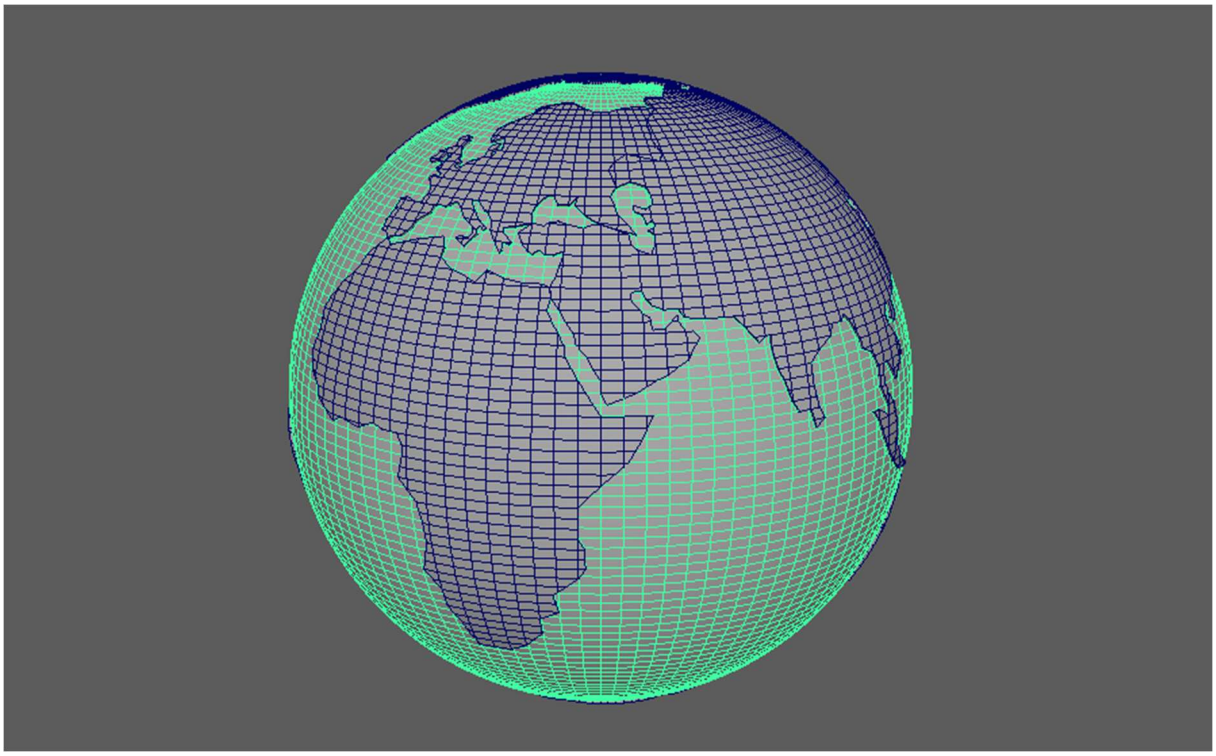
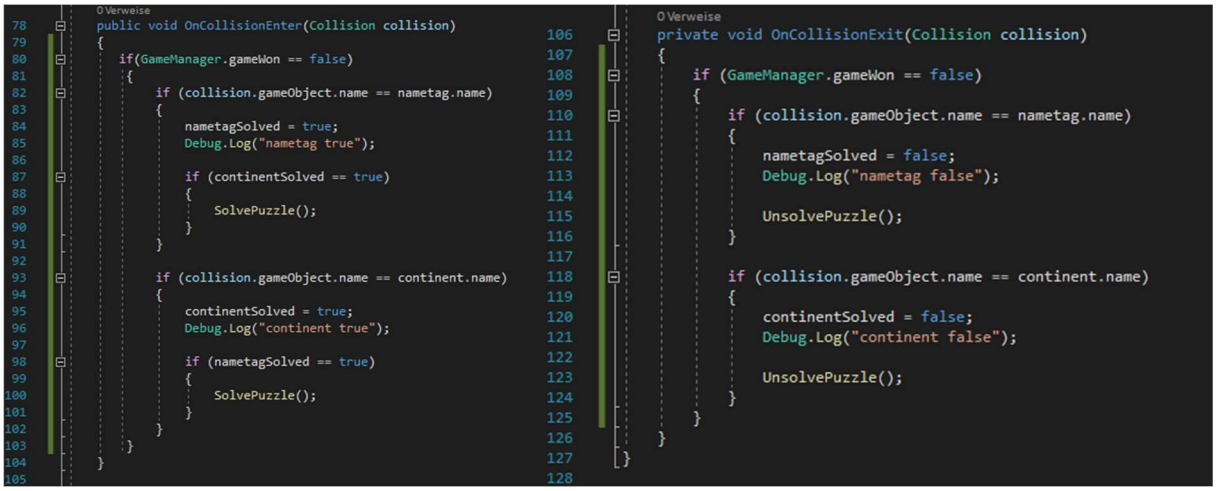


Abb. 3: Globus mit separaten Kontinenten in Maya

›Nun konnte mit der Programmierung losgelegt werden. Durch „OnCollisionEnter“ und „OnCollisionExit“ Methoden wird abgefragt, ob der Collider des dazugehörigen Namensschildes oder der des dazugehörigen Kontinents den Collider des Sockels berührt. Dies wird mit „GameObject.name“ der kollidierenden Objekte überprüft. Während sich sowohl das Namensschild als auch der Kontinent auf dem richtigen Sockel befinden werden die beiden Lichter über dem Tierbild mit „Light.color“ grün eingefärbt. So wird den Spieler*innen mitgeteilt, dass sie richtig liegen. Sobald alle 7 Puzzleteile gelöst sind und somit die 7 „puzzle#Solved“ bool-Variablen den Wert „true“ besitzen, ist das Spiel gewonnen und die Farbe des Hauptlichtes ändert seine Farbe von grün zu magenta mithilfe von „Mathf.PingPong“ und „Color.Lerp“ in einer Endlosschleife. Zusätzlich wird mit „GameObject.SetActive(true)“ der Text „You Won!“ über dem Globus eingeblendet.



```
0 Verweise
public void OnCollisionEnter(Collision collision)
{
    if(GameManager.gameWon == false)
    {
        if (collision.gameObject.name == nametag.name)
        {
            nametagSolved = true;
            Debug.Log("nametag true");

            if (continentSolved == true)
            {
                SolvePuzzle();
            }
        }

        if (collision.gameObject.name == continent.name)
        {
            continentSolved = true;
            Debug.Log("continent true");

            if (nametagSolved == true)
            {
                SolvePuzzle();
            }
        }
    }
}
```

```
0 Verweise
private void OnCollisionExit(Collision collision)
{
    if (GameManager.gameWon == false)
    {
        if (collision.gameObject.name == nametag.name)
        {
            nametagSolved = false;
            Debug.Log("nametag false");

            UnsolvePuzzle();
        }

        if (collision.gameObject.name == continent.name)
        {
            continentSolved = false;
            Debug.Log("continent false");

            UnsolvePuzzle();
        }
    }
}
```

Abb. 4: „OnCollisionEnter“ und „OnCollisionExit“ Methoden

Solange ein Kontinent nicht entnommen wird, besitzt der Rigidbody die Eigenschaft „Kinematic = true“. So fallen die Kontinente beim Start nicht direkt nach unten. Sobald der Kontinent vom Globus entfernt wird, wird die Kinematic Eigenschaft auf „false“ gesetzt, damit das Objekt von nun an ganz normal von der Gravitation beeinflusst wird. **– Zeitaufwand: 2,5h**

14. April 2020

Damit die Spieler*innen den Globus von allen Seiten, also auch von unten, sehen können, dreht sich dieser langsam in einer Endlosschleife. Dies wurde mit einer „Animator“ Komponente, einem Animation Controller und einem Animation Clip, indem die y-Rotation des Globusses animiert wird, umgesetzt. **– Zeitaufwand: 0,5h**

Als nächstes wurden Post-Processing Effekte zur Kamera hinzugefügt. Effekte wie „Ambient Occlusion“, „Bloom“ und „Color Grading“ wurden verwendet. Danach wurde die Licht-Intensität der einzelnen Lichter noch feinjustiert. **– Zeitaufwand: 0,5h**

Zusätzlich wurde das Readme mit neuen Screenshots ausgestattet.

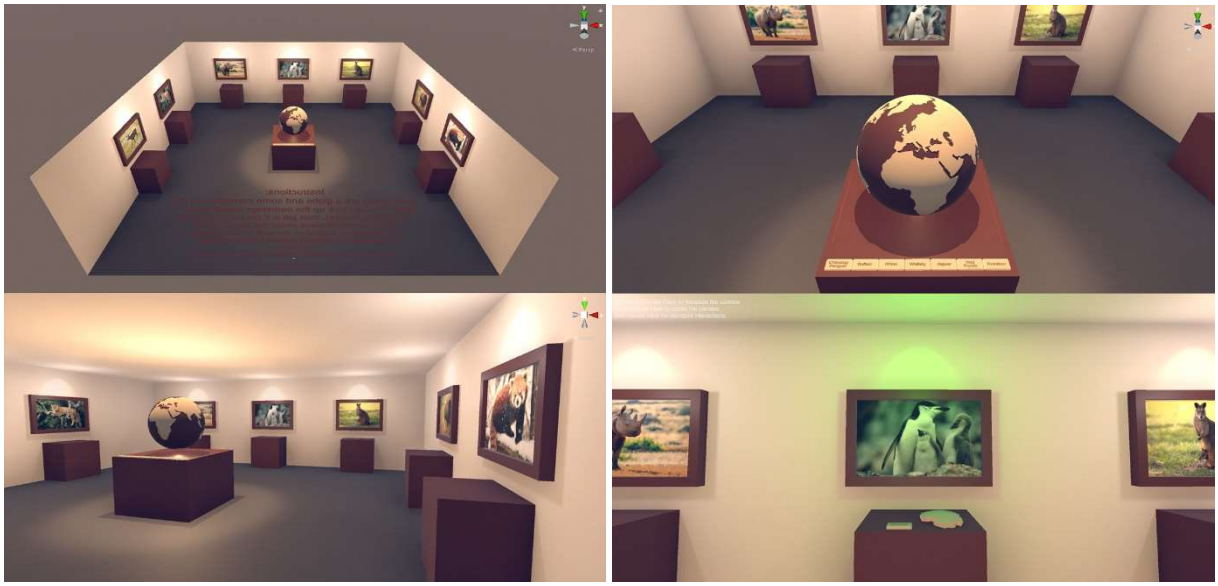


Abb. 4: Finaler Look der Unity-Szene

19. April 2020

Es wurden noch kleine Skript-Anpassungen vorgenommen. Sobald das Spiel gewonnen ist, sind alle Puzzles quasi eingeloggt, und selbst wenn man Kontinente oder Namensschilder wieder weggibt, bleibt das Spiel im gewonnenen Zustand. Außerdem wurden die werfbaren Objekte mit einer „Velocity Estimator“ Komponente ausgestattet. Dadurch können die Objekte beim Loslassen der Maus, je nach Geschwindigkeit verschieden weit geworfen werden. **– Zeitaufwand: 0,5h**

Zusätzlich wurden zwei Soundeffekte hinzugefügt. Ein „SolveSound“, der abgespielt wird sobald eins der 7 Puzzle gelöst wurde, und ein „WinSound“, der abgespielt wird sobald alle Puzzles gelöst wurden. Im Skript werden die Sounds durch AudioSource.PlayOneShot(AudioClip, volumeScale) aufgerufen. **– Zeitaufwand: 0,5h**

Zuletzt wurden finale Anpassungen im Readme vorgenommen, wie das Aktualisieren des Projektstandes und die Verlinkung der finalen Projektdokumentation.