

Fabrizio Neumeister

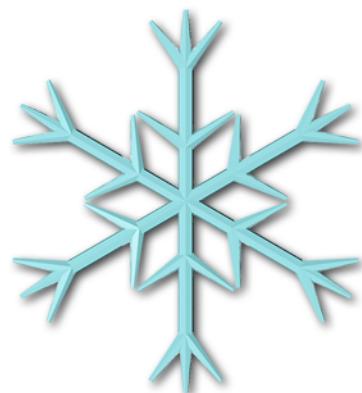
Matrikel-Nr.: 408002

Game Engineering Lab

Fakultät: Informatik - Game Engineering

"Adventure Island"

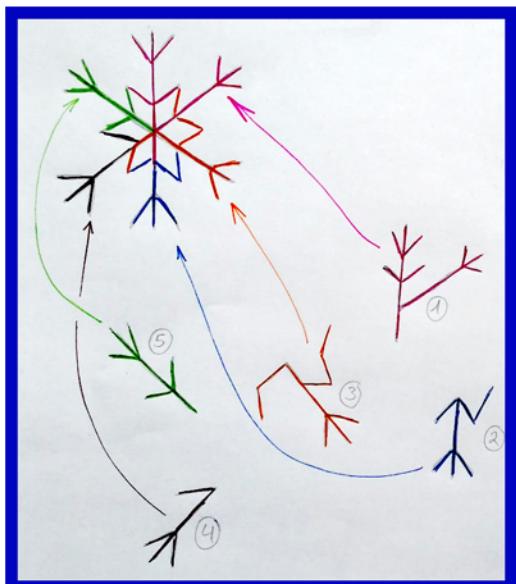
Winterinsel



# I. Konzeptidee

Da das Projekt "Adventure Island" auf 4 separate Inseln aufgeteilt wurde und die Winterinsel als letzte Hürde anzusehen ist, war die Überlegung, die Insel nach Beenden der Rätsel mit einem kleinen Boot zu verlassen. Um dies frostiger, dem Thema entsprechend zu gestalten, sollte der\*die Spieler\*in ein Schiffs-Steuer in der Form einer Schneeflocke finden.

Als Basis für die Form der Schneeflocke bediente ich mich an Ästen und deren Verzweigung und erstellte dazu eine Skizze, wie einzelne Elemente in orthographischer Sicht gewünschtes Element zeigen.



Skizze, Anordnung Äste zu Schneeflocke

Diese Skizze habe ich direkt übernommen und so in das Spiel implementiert. Da nun fünf einzelne Objekte (Äste) vorhanden waren, war die Idee, diese in weitere kleine Rätsel zu verpacken. Schließlich wurde sich in der Gruppe auf folgende Positionen geeinigt:

1. Ast - Hängt als Zweig an einem besonderen Baum mit Eislaub leicht versteckt auf der Insel
2. Ast - Ist in Eis gefroren, welches der\*die Spieler\*in erst auftauen muss
3. Ast - Verwendet als Teil des Gipfelkreuzes, welches nur durch das Lösen eines weiteren Rätsels erreicht werden kann
4. Ast - Liegt unter Schnee versteckt und muss ausgegraben werden, die Position verrät eine kleine Karte
5. Ast - Bildet den Arm eines Schneemanns, welcher als Dekoration zu sehen ist

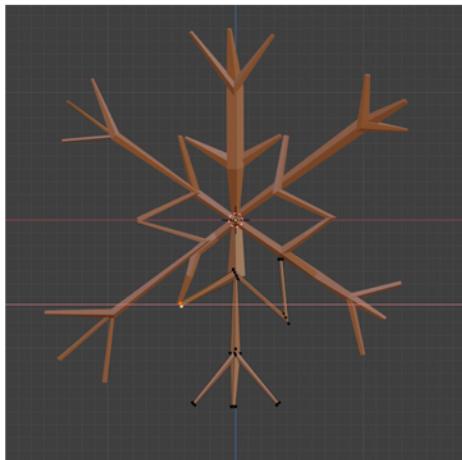
Sobald der\*die Spieler\*in alle Äste eingesammelt hat, kann er\*sie die Äste auf eine Art Podest/Altar legen. Durch ein Rotationsrätsel muss dann aus den Einzelteilen die Schneeflocke geformt werden, welche es dann als Belohnung und Schiffs-Steuer gibt.

## 2. Modellierung

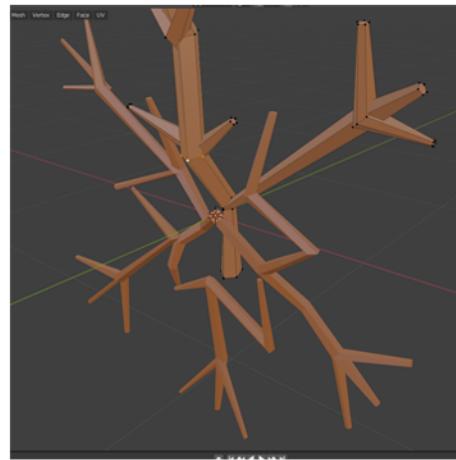
Die Äste wurden in Blender modelliert. Schwierigkeit dabei war, eine natürliche Form beizubehalten und sie von vorne betrachtet in richtiger Anordnung wie eine Schneeflocke aussehen zu lassen. Dazu bediente ich mich der orthographischen Ansicht in Blender und nahm meine Skizze als Referenzbild.

Ich begann für jeden Ast mit einem einzelnen Vertice, welchen ich dann entlang der Strecke des Astes immer aufs Neue extrudierte habe. Anschließend wandelte ich diesen Pfad in eine Curve um, damit ich dann mit einer weiteren Curve in Form eines Fünfecks, den Pfad entlang modellieren konnte. Dazu verwendete ich die Bevel-Eigenschaft der Curve-Funktion. Nach der Rückkonvertierung wurden die Vertices an den Enden der Zweige näher zusammengeschoben.

Anschließend wandte ich mich zur perspektivischen Ansicht und verschoß die Vertices an den Zweigpunkten der Äste so, dass jeder Ast eine natürlichere Form erhält.



Orthographische Ansicht in Blender

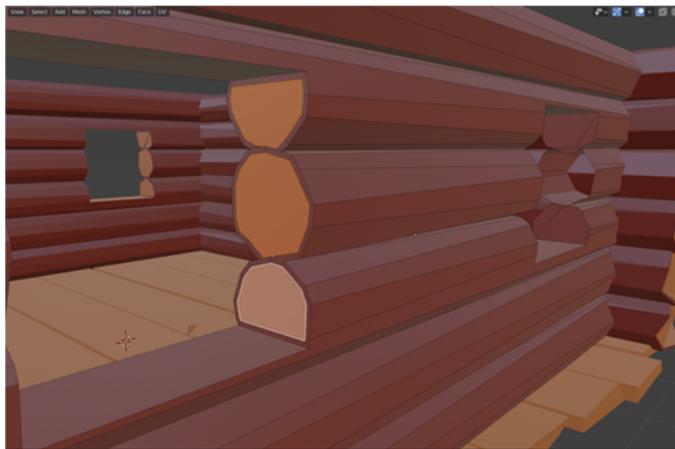


Perspektivische Ansicht in Blender (plastischer Ast)

Mit dieser Technik ist es mir möglich, bei korrekter Rotation eine Schneeflocke darzustellen und die einzelnen Äste in der Szene authentisch abzubilden. Durch einen einheitlichen Pivot-Punkt in der Mitte sind die Modelle auch bestens für das Rotationsrätsel geeignet.

Die Äste selbst bilden keine perfekte Symmetrie, um die Natürlichkeit zu bewahren. Die Schneeflocke wurde mit selber Methode als ein einziges Objekt neu modelliert. Dabei wurde auf die perfekte Symmetrie geachtet, indem ich nur einen Teil der Schneeflocke modellierte und dann mit dem Mirror-Modifier den Rest des Meshes vervollständigt habe.

Für das Ambiente entschied ich mich für eine gemütliche kleine Holzhütte, welches die kalte Winterinsel etwas lebendiger machen sollte. Für die Details der Hütte verwendete ich den Boolean-Modifier, um Abnutzungen im Holz zu generieren. Als Innenleben dient ein kleines Bett, zwei Schränke, ein Stuhl und ein Tisch, auf welchem später die Karte mit dem Hinweis für einen Ast lag. Die Hütte wurde auf dicke Stelzen gestellt mit einem kleinen Vorsprung, welchen der\*die Spieler\*in über eine kleine Treppe erreicht. Innen gibt es zwei Deckenlampen als Lichtquelle.



Modellierung der Fensteraushöhlung (Hütte) in Blender



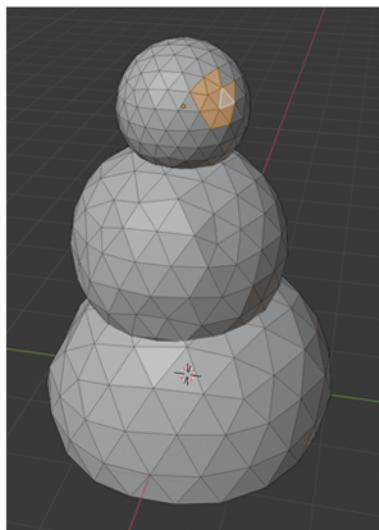
Modellierung des Innenlebens (Hütte) in Blender

Das Boot zum Verlassen der Insel wurde recht einfach gehalten. Über einfache geometrische Formen und einem Segel konnte später noch das Schiffs-Steuer angebracht werden.

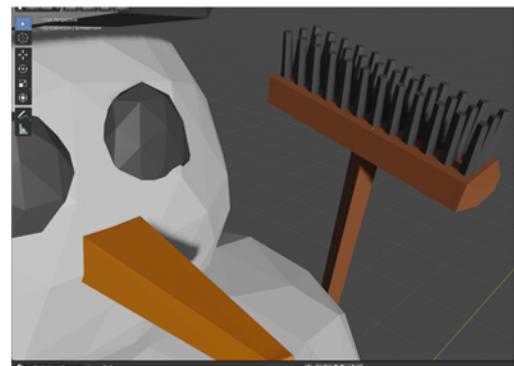
Der Schneemann wurde zuerst aus Kugeln geformt, bei welchen einzelne Faces gezielt verschoben wurden. Dies simuliert die natürliche Verformung des Schnees. Außerdem erhielt er einen Besen als Arm-Ersatz sowie einen Hut, zwei Steinen als Augen und eine Karotte als Nase. Das bringt der Szene Lebendigkeit und Freundlichkeit.



Das Boot in Blender



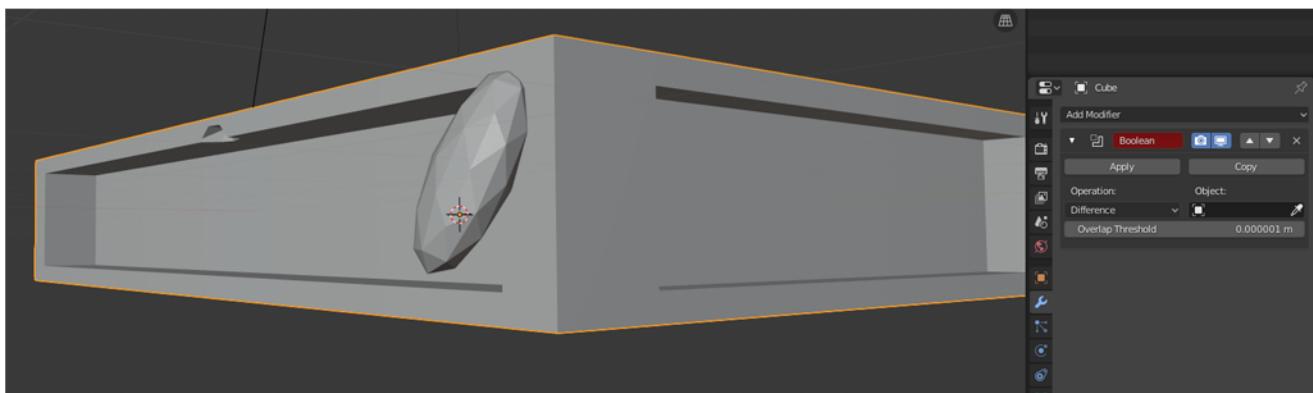
Die Kugeln vom Schneemann, einzelne Faces werden versetzt, sodass Schneeform entsteht



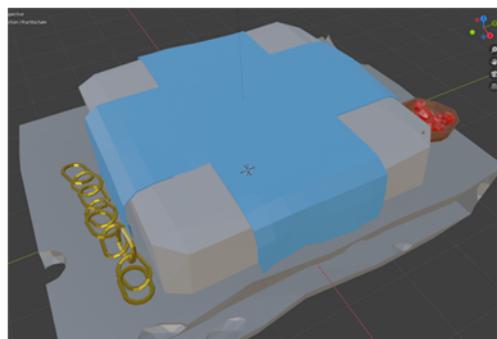
Schneemann-Gesicht mit Besen in Blender, der Ast befindet sich im Spiel später auf der anderen Seite. Er muss ein Einzelobjekt sein, da man den Ast später aufheben können muss.

Das Podest / der Altar habe ich ähnlich wie die Hütte mit dem Boolean-Modifier bearbeitet, damit dieser Einkärbungen und einen natürlichen Alterszerfall bekam. Eine steinerne Unterplatte ziert dieses Mesh, auf welches ich dann einen Eisblock modelliert habe, damit die Äste einen guten Kontrast zum Hintergrund aufweisen. Ursprünglich habe ich das Modell so erstellt, dass es von oben betrachtet werden konnte. Grund der Spielmechanik wurde das Podest / der Altar schräg aufgestellt, damit der\*die Spieler\*in einen besseren Blick beim Rotieren darauf hat.

Außerdem modellierte ich über einzelne Holzbretter und Stämme, die Vorlage entnahm ich der Hütte, einen Steg, damit der\*die Spieler\*in am Ende zum Boot gelangen kann.

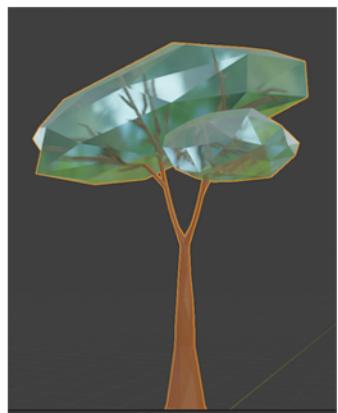


Der Altar, noch ohne Verformung, bei Verwendung des Boolean-Modifiers in Blender, ohne Textur

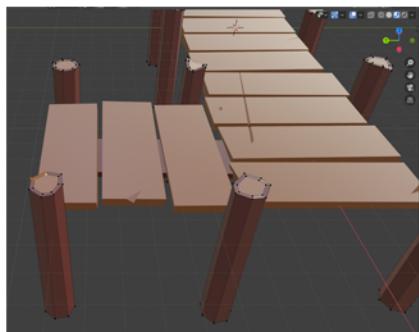


Podest/Altar in alter Optik in Blender

Der Altar wurde ursprünglich mit einem Tuch, einer Goldkette und einer hölzernen Fruchtschale versehen, um Details zu ergänzen. Die Holzschale wurde später eine Eisschale, die Goldkette zu Gunsten der Knöpfe für das Rotationsrätsel entfernt und das Tuch entfernt. Der Unterbau des Podestes/Altars wurde verschroben und belöchert, um weiteren zeitlichen Verfall darzustellen.



Baum mit Eislaub in Blender

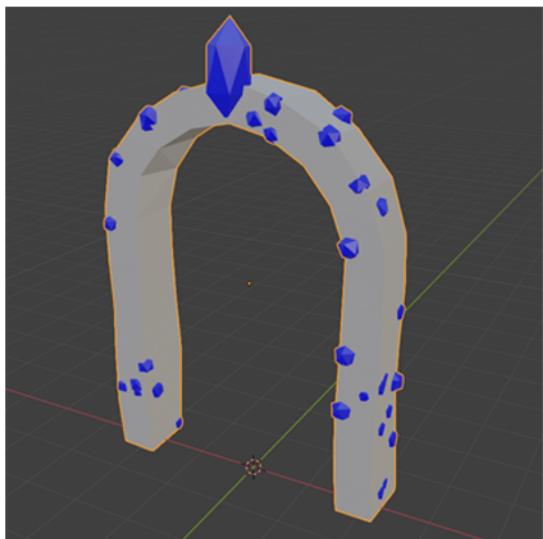


Steg für das Boot in Blender

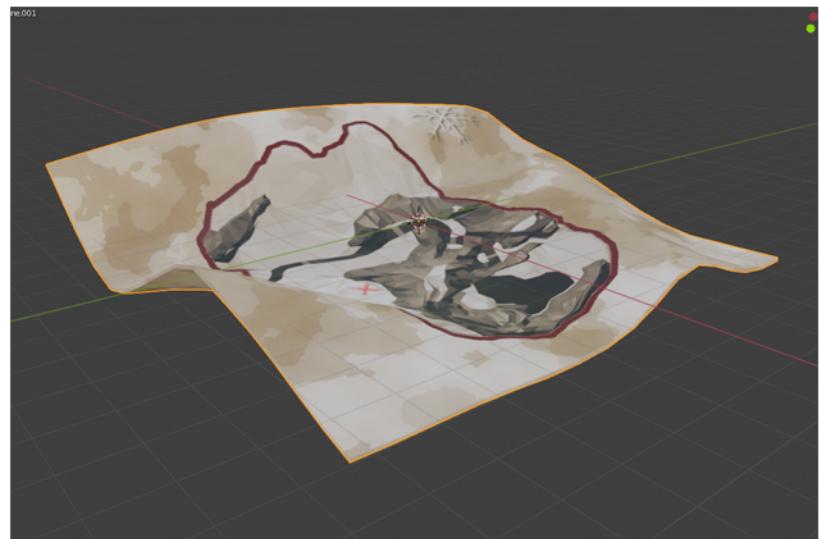
Da wir die Insel auch auf eine Art betreten müssen, wurde sich im Team auf Portale geeinigt, welche von der vorherigen Insel auf die aktuelle Insel führen. Um den Eingang zu unseren Insel zu simulieren modellierte ich ein einfaches steinernes Portal mit kristallartigen Eisklötzen und einer Art magischen Eiskristall auf der Spitze. Dazu erstellte ich einen Halbkreis und extrudierte die unteren Vertices weiter hinunter. Wie bei den Ästen bediente ich mich dann über die Curve-Funktionen der Bevel-Eigenschaft.

Die Eis-Kristalle wurden später mittels dem Weight-Paint und einem Particle-System in zufälliger Reihenfolge auf das Portal modelliert.

Die Karte mit dem Hinweis für den vergrabenen Ast habe ich zuerst als Plane realisiert, welche dann den Cloth-Modifier bekam. Mit Hilfe von weiteren Objekten in Blender ließ ich dann per Animation dieses Mesh fallen, sodass es eine natürliche Verformung annehmen konnte. In Photoshop erstellte ich über einen orthographischen Screenshot der ersten Version der Insel ein Bild für die Schatzkarte, welches ich dann als Image-Texture auf die Karte gelegt habe.



Portal in Blender



Hinweiskarte in Blender, verformt durch Cloth-Animation

# 3. Probleme der Modellierung

Beim Modellieren bereits kommt es immer zu kleinen unerwarteten Problemen. Diese und deren Behebung möchte Ich kurz aufführen:

## 1.) Absprache mit dem Team

Um einen einheitlichen Look gewährleisten zu können war es notwendig, Absprache mit dem Team zu halten, da jeder für eigene Objekte verantwortlich war. Unter den Einigungen war unter Anderem Thema, dass wir für Objekte wie Äste, Stämme und generell längliche Dinge ein Fünfeck oder Siebeneck als Vorlage nehmen. Das behält den Low-Poly-Look bei und sorgt Grund der ungeraden Zahl auch für mehr Individualität. Weiterhin wurden die Farbtöne für z.B. Hölzer in einer Team-Sitzung besprochen, damit alles einen einheitlichen Stil bildet. Wir einigten uns zudem darauf, keine vorgefertigten Meshes zu verwenden und teilten im Sinne der Rätsel jedem Objekte zu.

## 2.) Verwendung in Unity

Ich habe alle Modelle in Blender komplett erzeugt und auch bereits mit einem Material versehen, stellte dann aber fest, dass Unity eine andere Renderart benutzt. So habe ich jedes Modell in Unity von seinem ursprünglichen Material trennen müssen, nachdem es importiert wurde, um dort den Materialien neue Eigenschaften zuweisen zu können. Ein Vorteil, der sich daraus bot war, dass ich die Zahl der Materialien reduzieren konnte, da wir jedes Objekt einzeln in einer Szene in Blender modelliert hatten. Trotz zeitintensiver Bearbeitung sieht es in Unity noch immer anders als in Blender aus, dennoch aber für sich selbst zufriedenstellend und passend.

Ein weiteres Mal musste ich alle Materialien überarbeiten, als sich das gesamte Team aller Inseln dazu entschied, auf die "Universal Render Pipeline" umzusteigen. Trotz Funktion ließen sich die Materialien nicht wie zuvor darstellen.

## 3.) Proportionen, Skalierungen und Ausrichtungen

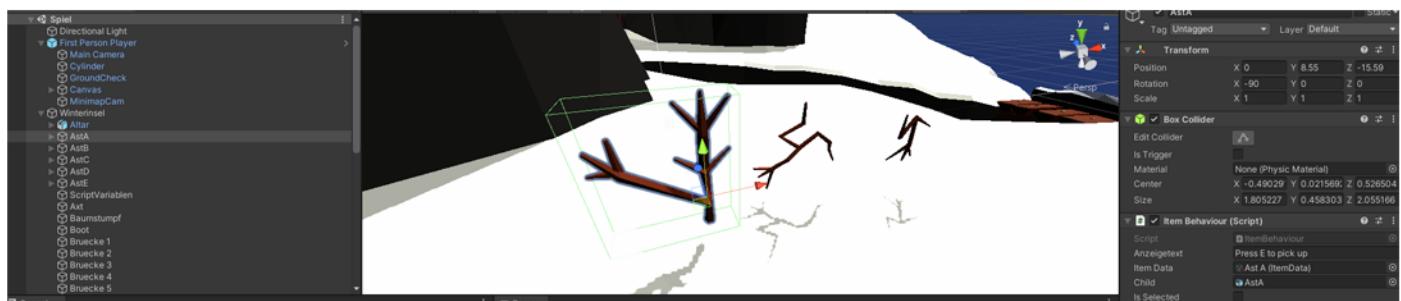
Viel Zeit verschlang ebenfalls das Anpassen aller einzelnen Modelle an eine Gesamtumgebung. In Unity müssen einzelne Objekte aufhebbar sein, da sie dem Inventar zugefügt werden können, weshalb sie als "Child" eines "Empty" in die Szene geladen werden müssen. Das verändert deren relative Größe im Spiel zum Original, welches aber später eventuell wieder in die Spielszene instanziert werden muss. Die Skalierung der Insel, die Proportionen der Hütte und ihrer Türe, der Tisch und die Stühle, die Bäume und mehr mussten auch auf den Spieler in der Szene angepasst werden. Für die Darstellung der Äste auf dem Podest/Altar musste auch hier mit der Größe etwas experimentiert werden.

## 4. Implementierung des Codes

Bei der Programmierung kümmerte ich mich selbständig um das Rotierrätsel mit den Ästen und der Schneeflocke. Dabei mussten einige Hürden Stück für Stück bearbeitet werden, sodass am Ende alles funktioniert hat.

### 1.) Einarbeitung in das Inventar-Skript der anderen Gruppe

Die Herbstgruppe hat für uns das "Item-Behaviour" Skript angelegt. Damit war es möglich, einzelne Objekte aufzuheben und seinem Inventar zuzufügen. Ein paar der vorhandenen Methoden habe ich für eigene Skripte übernommen und so modifiziert, dass ich mein Rätsel realisieren kann. Zuerst galt es, alle benötigten Objekte anzulegen, die man aufheben kann und alle, die statisch sind. Zu Testzwecken wurden diese dann nebeneinandergestellt, um im laufenden Spiel testen zu können.



Drei der fünf Äste zusammen in der Szene in Unity, zum schnellen Auffinden

### 2.) Hilfs-Variablen anlegen

Das Anlegen eines "Empty"-Objektes mit einem Skript voller Variablen war notwendig, um von verschiedenen Skripten auf die Daten zugreifen zu können. So kann man z.B. abfragen, welcher Ast bereits auf dem Altar liegt oder wie viele Äste gesammelt wurden.

### 3.) Skript beginnen

"ItemData" anlegen: Jeder Ast benötigt eine eigene Zuweisung als Item für die nötigen Daten.  
 "GameObject" anlegen: Jeder Ast wird aus dem Inventar auf den Altar neu instanziert, sodass jeder Ast ein neues GameObject wird. Diese Variable ist die Zuweisung dafür, außerdem musste ich den Altar selbst als "Transform" speichern, um den neuen Ästen später den Altar als Parent zuweisen zu können. Das Skript für die Rotation der Äste, welches später implementiert wurde, wird in das Skript geladen und gestartet, sobald alle Stecken drauf sind.

Sobald der\*die Spieler\*in zum Ast läuft, wird mittels "Raycast" und der Tasteneingabe "E" geprüft, welchen Ast der\*die Spieler\*in in der Hand hält. Dieser wird aus dem Inventar gelöscht und neu als "Child" vom Altar in das Spiel geladen. Er erhält eine zufällige Rotation entlang der Y-Achse. Diese Rotation muss separat für jeden Ast in einer Variable gespeichert werden, da es in Unity Komplikationen mit den realen "transform"-Daten gab.

Jeder Ast, der so auf den Altar gelegt wurde, erhält nun noch einen "MeshCollider", bei welcher die "Convexe Hülle" angeschaltet wurde. Somit wird sichergestellt, dass der\*die Spieler\*in später bequem einzelne Äste per Raycast trifft und auswählen kann. Zusätzlich benötigt jeder Ast das Skript, das seine Auswahl ermöglicht. Der zuletzt aufgelegte Ast ist der aktive Ast, dessen "Bool-Wert" in dem Hilfsvariablen-Skript steckt. Zuletzt wird geprüft, ob nach dem Auflegen des aktuellen Astes alle Äste auf dem Podest/Altar vorhanden sind. Falls dem so ist, wird der BoxCollider vom Altar gelöscht und dessen Skript entfernt. So kollidieren die Boxen nicht beim Auswählen der Äste.



Podest/Altar im Testspiel, Ast "B" wurde markiert, Prüfung der Funktionalität des Codes

## 4.) Rotations-Rätsel schreiben

Das Rotationsskript bekam ein weiteres Skript mit Hilfsvariablen zugewiesen, da es während der Implementierung immer wieder Fehler gab. Durch das Zuweisen der Skripte erbten die Pfeilknöpfe eigene Rotationen, weshalb jeder Pfeil eine unabhängige Zuweisung bekam. Durch das "Auslagern" auf ein separates Skript wurde die mehrfache individuelle Wertzuweisung verhindert. Deshalb wurden zwei Skripte mit Hilfsvariablen in den Code integriert.

Nun setzte ich zwei "Vector3"-Datentypen, einmal auf (0, 0, 0) und einmal einen angepassten Wert. Die fertigen Stecken, sobald das Rätsel gelöst wurde, "lerpen" zu einem Punkt-Objekt und verschwinden dann und aus der Mitte heraus "wächst" dann die Schneeflocke als Preis.

Beim Drücken von "E" auf eine der Pfeiltasten überprüft das Skript zuerst, welcher Ast aktiv gewählt ist und rotiert diesen dann um 10 Grad in gewünschte Richtung. Außerdem setzt er den aktuellen Wert der Rotation entweder +10 oder -10 des jeweiligen Astes. Sollte dieser Wert 360 erreichen, wird er automatisch auf 0 gesetzt und unter 0 automatisch auf 350. So kann ich nun nach jeder Rotation anhand der fünf Variablen überprüfen, ob alle denselben Wert haben. So kann die Schneeflocke selbst egal in welcher Rotation liegen, damit das Rätsel gelöst wird.

Sobald das Rätsel gelöst ist, zeigt sich kurz die Animation und am Ende bekommt die Schneeflocke das "ItemBehaviour"-Skript mit einem BoxCollider, womit man es, wie die einzelnen Äste zuvor, aufheben kann.

Die Zuweisung der Farbe eines Astes ist in Unity nur möglich, wenn ich nicht nur die Zuweisung des Materials ändere, ferner muss ich ein ganzes Material übergeben. Das Material muss vorher als solches zwischengespeichert sein. Hierbei bekam ich Hilfe unseres Dozenten, da stundenlange Recherche nicht zum gewünschten Ergebnis führten und ich die Fehlerquelle nicht auffand. Unity speichert für jedes Objekt zudem die Daten der "transform" unabhängig vom Wert, welcher im "Inspector" angezeigt wird. Deshalb war eine weitere Herausforderung zu tüfteln, wie ich die komplexe mathematische Umrechnung vereinfachen kann und fand heraus, dass meine Objekte als "Child" dennoch mit der Rotationsfunktion um "menschlich lesbare Werte" rotiert werden konnte. Es wurden deutlich mehr Variablen benötigt, als anfangs gedacht, wie zum Beispiel die Überprüfung, ob alle Äste bereits auf dem Podest/Altar liegen, ob das Rätsel bereits gelöst wurde oder ob das Lerpen bereits zu Ende ging.

Insgesamt kommt der Code, mit Kommentaren zur Erläuterung und auf vier Skripte aufgeteilt, auf insgesamt 559 Zeilen. Nach der Implementierung des Codes konnte ich die Äste in der Spielszene den anderen Teamkollegen mitgeben, sodass diese ihren jeweiligen Ast für sich in der Szene wie gewollt platzieren konnten.



Test des Rätsels, ob Sichtbarkeit der Schneeflocken-Form gewährleistet ist sowie weitere Funktionstests.



Test des kompletten Rätsels abgeschlossen,  
Spieler\*in hält am Ende Schneeflocke im  
Inventar.

```
        }
    }
    else if (AstC == Inventory.instance.getSelectedItemObject().GetComponent<ItemBehaviour>().itemData)
    {
        Inventory.instance.getSelectedItemObject().GetComponent<ItemBehaviour>().RemoveFromInventory();
        Instantiate(astC, AltarPunkt.position, AltarPunkt.rotation);
        C = true;
        GameObject.Find("AstC(Clone)").transform.SetParent(AltarPunkt);
        GameObject.Find("AstC(Clone)").transform.Translate(0, 0.9f, 0);
        int zufallsZahl = Random.Range(1, 35) * 10;
        GameObject.Find("AstC(Clone)").transform.Rotate(0, zufallsZahl, 0);
        rota.rotC = zufallsZahl;
        Debug.Log(zufallsZahl + " und " + rota.rotC);
        MeshCollider mcc = GameObject.Find("AstC(Clone)").AddComponent<MeshCollider>() as MeshCollider;
        mcc.convex = true;
        GameObject.Find("AstC(Clone)").AddComponent<AstRaetsel>();
        GameObject.Find("AstC(Clone)").GetComponent<AstRaetsel>().anzeigetext = "Mark Branch C";
        if ((A && B) && (D && E))
        {
            GameObject.Find("Rechtspfeil").AddComponent<AstRaetsel>();
            GameObject.Find("Rechtspfeil").GetComponent<AstRaetsel>().anzeigetext = "Rotate clockwise";
            GameObject.Find("Linkspeil").AddComponent<AstRaetsel>();
            GameObject.Find("Linkspeil").GetComponent<AstRaetsel>().anzeigetext = "Rotate counter-clockwise";
            auswahlC = true;
            Destroy(GameObject.Find("Altar").GetComponent<MeshCollider>());
            Destroy(this);
        }
    }
    else if (AstD == Inventory.instance.getSelectedItemObject().GetComponent<ItemBehaviour>().itemData)
```

Code-Ausschnitt aus "Visual Studio", hier die Abfrage nach "Ast C" im Inventar, um ihn mit zufälliger Rotation auf den Altar zu legen inklusive Abfrage, ob alle Äste bereits vorhanden sind.

## 5. Grafik-Arbeiten

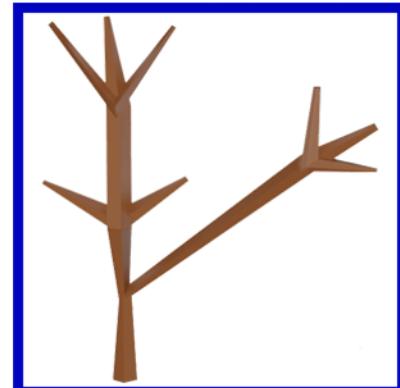
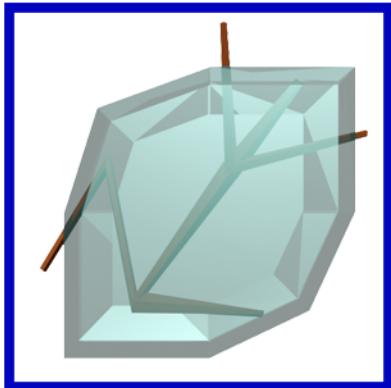
Ich durfte einige kleinere Grafik-Arbeiten für die Gruppe leisten. So habe ich zum Beispiel dafür gesorgt, dass im Inventar jedes unserer Meshes als Sprite korrekt angezeigt wird.

Hierzu bekam ich die Modelle aller zugeschickt, habe sie in Blender geöffnet und für jedes einzelne Mesh, was der \*die Spieler\* in im Spiel aufheben kann, eine kleine Szene geschaffen. Eine Plane im Hintergrund mit einem Emitter-Material sorgte für einen neutralen Hintergrund, den ich später in Photoshop benötigt habe.

Beim Beleuchten deaktivierte ich Schatten, sodass ich ein Bild ohne Fremdfragmente erstellen konnte. Per orthographischer Kamerasisicht renderte ich dann Bilder der einzelnen Modelle raus und bearbeitete sie in Photoshop weiter.

Durch den neutralen Hintergrund konnte ich diesen entfernen, wandelte das Bild in ein PNG mit der Eigenschaft "interlaced" um, um eine Transparenz im Sprite zu ermöglichen.

Für die gesamte Gruppe durfte ich für die Präsentation einen allgemeinen schönen Rahmen basteln. Dabei habe ich versucht, alle Jahreszeiten und deren typischen Pastell-Farben zu integrieren.



Beispiele von Inventar-Sprites, links: Ast im Eis, mitte: Schaufel zum Graben, rechts: loser Ast



Außerdem durfte ich für unsere Gruppe die Skybox in Unity kreieren. Diese entfaltet ihre ganze Schönheit im Spiel selbst, deshalb hier ohne Screenshot.

Design für die Präsentation, links oben: Frühling, rechts oben: Sommer, links unten: Herbst, rechts unten: Winter.

Typische Pastell-Farben für die Jahreszeit und einzigartige Symbolik der Jahreszeit integriert.

## 6. Quellen

### Meshes:

Alle Meshes wurden ohne jegliche Vorlage/Vorgabe oder Ähnlichem von mir selbst erstellt. Es gab keine weitere Person, die in irgendeiner Art beteiligt gewesen ist.

### Materials:

Alle Materials wurden ohne jegliche Vorlage/Vorgabe oder Ähnlichem von mir selbst erstellt, da alle Objekte bis auf die Hinweiskarte aus einzelnen Farben bestehen. Die Image-Textur der Hinweiskarte wurde selbst erstellt.

### Images:

Alle Images wurden ohne jegliche Vorlage/Vorgabe oder Ähnlichem von unserem Team selbst erstellt. Meine Vorlagen waren lediglich die Meshes meiner Team-Mitglieder. Alle Symbole wurden selbst von mir erstellt (Schneeflocke, Schmetterling, Sonne und Herbstblatt) sowie das Design für die Dokumentation (eigene Dokumentation sowie Gruppenpräsentation)

### Weitere Quellen:

Für den Umgang mit Unity habe ich mir den Kurs von "Prof. Dr. René Bühling" auf Udemy angesehen:

<https://www.udemy.com/course/csharp-gamedev/>

Zur Lösungsfindung einiger Probleme in Unity suchte ich das offizielle Forum der Game-Engine auf:

<http://www.forum.unity.com/>

Zum Verständnis von Unity-Funktionen half mir die offizielle Dokumentation von Unity:

<https://docs.unity3d.com/>