

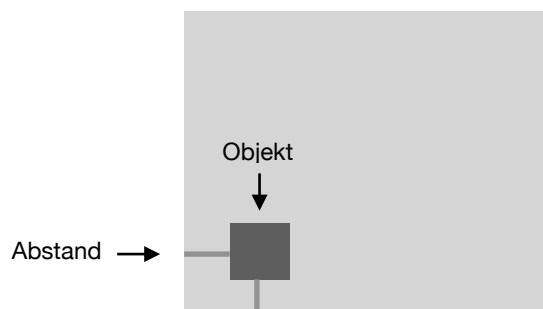
# PLOOM!

Plan your Room

## 1.1 Berechnung des Mindestabstands

Für die gleichmäßige Distribution jeweiliger Objekte in einem Raum, mit der zusätzlichen Berücksichtigung von einem Mindestabstand, ist die folgende Berechnung zuständig. Hierbei hat der Benutzer die Möglichkeit die Raumbreite, die Objektbreite und den Abstand zwischen jeweiligen Objekten manuell zu bestimmen. Raummaße werden dabei metrisch bestimmt, während Objekt und Abstand in Zentimeter angegeben werden müssen. Um der Kalkulation der ebenmäßigen Objektverteilung folgen zu können, wird die Berechnungsprozedur im Folgenden anhand von skizzierten Grafiken aufgeführt. Nach Angabe der Raum- und Objektbreite muss die Anzahl an verfügbarer Objekte in einer Zeile vorerst festgelegt werden. Da der Abstand zwischen den Objekten jedoch beachtet werden muss, ist für die Bestimmung der Objektanzahl in einem vorgegeben Raum folgende Berechnung notwendig:

$$\text{Objektanzahl} = \text{Raumbreite} : (\text{Objektbreite} + \text{Abstand})$$



Für diese Berechnung werden die Werte Objektbreite und Abstand, die durch den Nutzer an das System gelangen, jeweils addiert und anschließend die Raumbreite durch den neuen Wert dividiert und abgerundet. Die Abrundung verhindert das Abschneiden von Objekten, da sie als Ganzes erhalten bleiben sollten. Aus dieser Berechnung ergibt sich zudem ein **REST**-Wert, der in der finalen Berechnung für die Ermittlung des Mindestabstands inkludiert wird.



$$\text{Mindestabstand} = (\text{Objektanzahl} * \text{Abstand} + \text{REST}) : (\text{Objektanzahl} + \text{REST})$$

Um diese Berechnung in C# anhand von Unity umsetzen zu können, wird folgender Code benötigt:

```
public float AbstandBerechnen(float rauml, float objektl, float mindestabstand)
{
    int anzahl = (int)(rauml / (objektl + mindestabstand));
    float rest = rauml % (objektl + mindestabstand);
    float abstand = (anzahl * mindestabstand + rest) / (anzahl + 1);
    return abstand;
}
```

Die Funktion `AbstandBerechnen` führt anhand der Eingaben des Benutzers die oben aufgeführten Berechnungen durch, um den Mindestabstand zwischen den Objekten zu ermitteln.

## 1.2 Der Algorithmus

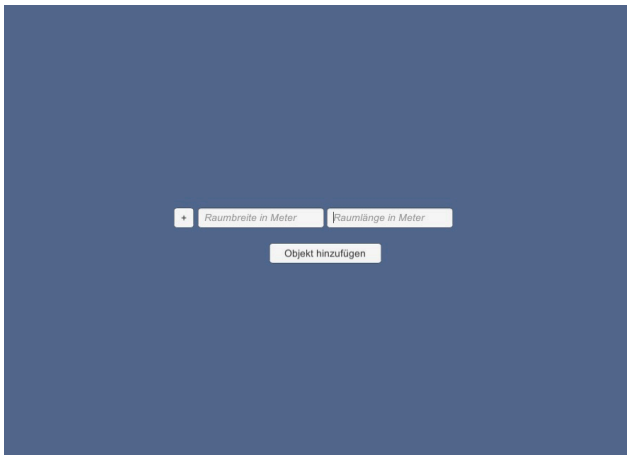
Um die Berechnungen auf sauberer Basis in das Programm implementieren zu können, wird ein Algorithmus benötigt, der diese Bedingung erfüllt. Hierfür implementierten wir zwei, ineinander verschachtelte While Schleifen, die anfangs den Raum anhand der Breite gefüllt, und dann jeweils ein Maß weiter in die Länge des Raumes geht, bis der ganze Raum gefüllt ist.

```
public void ObjektDraw()
{
    float x = 0;
    float y = 0;
    float abstandx = AbstandBerechnen(fbreite, obreite, oabstand);
    float abstandy = AbstandBerechnen(flaenge, olaenge, oabstand);

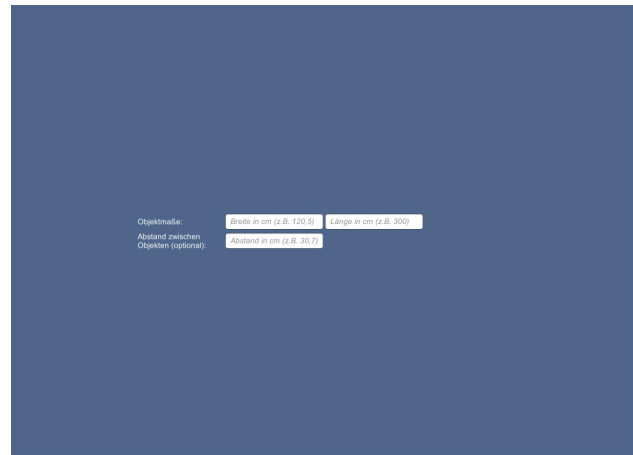
    while (y + abstandy + olaenge <= flaenge)
    {
        float centery = y + abstandy + (olaenge / 2);

        while (x + abstandx + obreite <= fbreite)
        {
            float centerx = x + abstandx + (obreite / 2);
            cube = GameObject.CreatePrimitive(PrimitiveType.Cube);
            cube.transform.position = new Vector3(centerx, 0.6f, centery);
            cube.transform.localScale = new Vector3(obreite, 1, olaenge);
            x = x + abstandx + obreite;
        }
        x = 0;
        y = y + abstandy + olaenge;
    }
}
```

## 1.3 Nutzereingaben

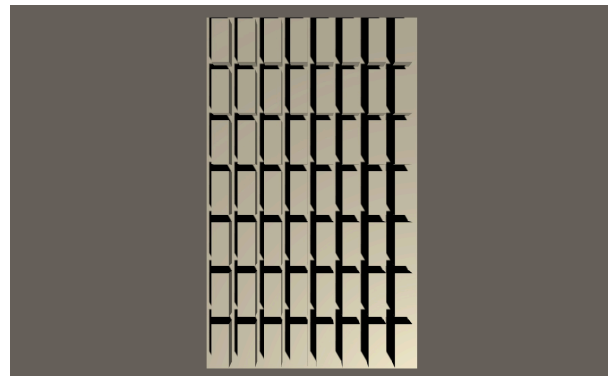
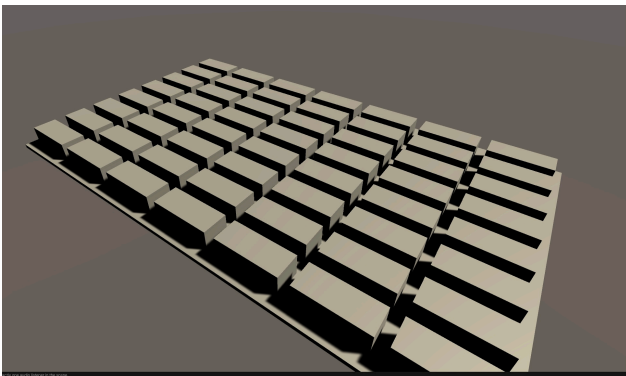


A screenshot of a user input form on a dark blue background. It features two input fields: 'Raumbreite in Meter' and 'Raumlänge in Meter'. Below these fields is a button labeled 'Objekt hinzufügen'.



A screenshot of a user input form on a dark blue background. It includes input fields for 'Objektmaße' (Object dimensions) with sub-fields for 'Breite in cm (z.B. 120,0)' and 'Länge in cm (z.B. 300)', and an optional field for 'Abstand zwischen Objekten (optional):' with a sub-field 'Abstand in cm (z.B. 30,0)'. There is also a button labeled 'Objekt hinzufügen'.

Der Nutzer wird anfangs nach der Raumbreite und der Raumlänge in Form einer metrischen Angabe gefragt. Nachdem er dies eingetragen hat, kann er die Dateneingabe fortsetzen indem er ein Objekt hinzufügt. Das Objekt wird hierbei in Form von einer Zentimeterangabe eingegeben. Nachdem die Objektbreite und Länge eingetragen wurde, kann der Abstand zwischen den Objekten ausgefüllt werden. Dieses Feld ist jedoch optional, da Objekte auch ohne Abstand auf einem Raum verteilt werden können. Wichtig zu beachten ist, dass dies nicht das Endprodukt darstellt, sondern nur einen Prototypen des PLOOM!-Systems.



Die unterschiedlichen Perspektiven der Ausgabe können ebenfalls vom Nutzer angepasst werden. Hierbei stehen ihm die 3D- sowie die Vogelperspektive zur Verfügung.

## 1.4 Anhang

# Notizen

Proof of Concept PLOOM!

Risiken:

1. Algorithmus zur Verteilung der Objekte in einem Raum  
Anpassen des Algorithmus zum Einsetzen von Hindernissen (Wege, Bühnen...)
2. Parametrisierung der Objekte in Blender
3. 2D vs. 3D

### 1. Lösungsansatz Algorithmus + Hindernisse:

Ein Algorithmus zur Verteilung von Objekten in einem Raum muss verschiedene Formen und Größen von Objekten sowie Räumen berücksichtigen.

Einfacher Algorithmus:

- Die Maße eines Raumes werden eingegeben. (Länge und Breite).
- Die Maße eines Objektes werden eingegeben. (Länge und Breite).
- Abstand zwischen Objekten eingeben.
- Abstand wird zu der Objektgröße hinzugerechnet.
- Der Flächeninhalt des Raums wird ausgerechnet.
- Der Flächeninhalt des Objektes & Abstand wird ausgerechnet.
- Teilen des Raumflächeninhalts durch Objektflächeninhalt (Anzahl Objekte, die in Raum passen).
- Den Raum in Zeilen aufteilen, die so groß sind wie die Länge des Objektes und die Breite des Raums.
- Objekte in Zeile füllen bis jeweils die Breite des Raums erreicht wurde. Dann zur nächsten Zeile springen und ebenfalls auffüllen, bis Länge des Raumes erreicht wurde.

Komplexer Algorithmus:

- Option zwischen verschiedenen Raumformen (Kreis, Rechteck)
- Bei Kreis: Raum, Objekt & Abstand als Kreisflächeninhalt berechnen. (Objektbreite/2 + Abstand = Radius)
- Einfüllen der Objekte basierend auf Flächeninhalt der Kreise

- Bei Rechteck: evtl. Raum unterteilen in mehrere Rechtecken. (z.B. L-förmiger Raum)
- Einfüllen der Objekte in die einzelnen Rechtecke

Komplexer Algorithmus + Hindernisse:

- Form des Hindernisses angeben z.B. Kreis (Säule), Gehweg (Rechtecke)
- Flächen die freigehalten werden sollen definieren via Koordinaten (Länge und Breite des Raumes in dem jeweiligen Rechteck/Kreis)
- Bei Kreis (Position der Säule (X, Y) + Umfang)
- Bei Rechteck (Position der Ecken des Gehweges via Koordinaten (X, Y), Gehwege in mehrere Rechtecke unterteilen, wenn z.B. L-förmig)

Nutzereingaben:

Raum Form (Kreis, Rechteck)

- Kreis: Durchmesser oder Radius oder Fläche angeben (Meter)

- Objektbreite & -länge eingeben (Centimeter)
- Objektabstand eingeben (Centimeter)
- Hindernisform wählen (Kreis, Rechteck)

2D Ansicht des Raumes zeigen

- Kreis: Koordinaten eingeben X-Achse und Y-Achse der Säule + Umfang (Meter)
- Rechteck: Koordinaten eingeben X-Achse und Y-Achse jeder Ecke. Bei L-Form in mehrere Rechtecke unterteilen (Meter)

- Berechnen Schaltfläche auswählen

- Rechteck: Bei L-Form in mehrere Rechtecke unterteilen. Breite & Länge oder Fläche jedes Rechtecks angeben. (Meter)

- Objektbreite & -länge eingeben (Centimeter)
- Objektabstand eingeben (Centimeter)
- Hindernisform wählen (Kreis, Rechteck)

2D Ansicht des Raumes zeigen

- Kreis: Koordinaten eingeben X-Achse und Y-Achse der Säule + Umfang (Meter)
  - Rechteck: Koordinaten eingeben X-Achse und Y-Achse jeder Ecke. Bei L-Form in mehrere Rechtecke unterteilen (Meter)
- Berechnen Schaltfläche auswählen
- Zwischen 2D und 3D Ansicht wechseln via Schaltfläche
- Rein und Rauszoomen via Fingerwisch
- Neue Berechnung auswählen