Game Programming Diploma
Praktische Übung 2 für Praxis Test 5

Grid Calc

Schwierigkeitsgrad

2 - 3

Zeitrahmen

2 Stunden

Lernziele

Grundlagen des Visual Scripting erlernen & Einarbeitung in Unreal

Beschreibung des Übungsablaufs

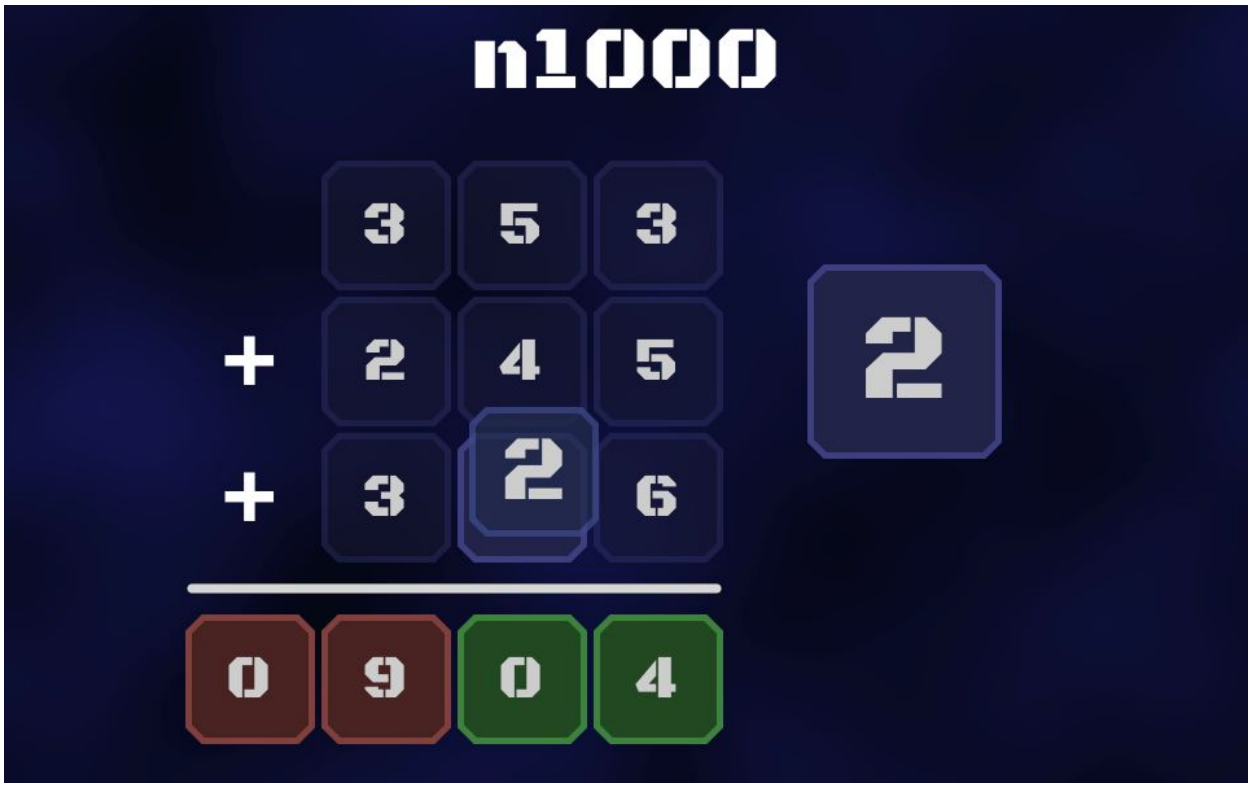
- Erstelle zuerst alle benötigten Blueprints
    - GameMode
    - GameState
    - PlayerController
    - Game-Widget
    - NumberItem-Widget
    - erbend von NumberItem-Widget:
        - AutoGetNumberItem-Widget
        - TotalDigitNumberItem-Widget
- Lasse beim Start des GameMode eine Instanz des GameWidgets erzeugen und dem Viewport hinzufügen.
- Die Maus soll während des Spiels angezeigt werden.
- Erstelle zwei Arrays (Integer, Boolean) mit je neun Elementen und einen Integer zum Halten der aktuellen Nummer im GameState
- Erstelle eine Funktion “IsSet”, die das Element im Boolean-Array zurückgibt.
- Erstelle eine Funktion “GetNumber”, die das Element im Integer-Array zurückgibt.
- Erstelle eine Funktion “GetCurrentNumber”, die die aktuelle Zahl zurückgibt - die Variable dafür soll privat sein.
- Erstelle eine Funktion “SetRandomNumber”, die die aktuelle Zahl zufällig auf einen Wert zwischen 1 - 6.
- Erstelle eine Funktion “SetNumber”
    - Diese soll prüfen, ob an der aktuellen Stelle bereits eine Zahl gesetzt ist. Wenn dies zutrifft, soll die Ausführung der Methode beendet werden.
    - In jedem anderen Fall, soll der Wert “True” in das Boolean-Array und die aktuelle Zahl in das Integer-Array gefüllt, und anschließend eine neue, zufällige Zahl erzeugt werden.
- Erstelle eine Funktion “GetTotalNumber”, die die Summer aller Elemente im Array zusammenrechnet.
    - Die Power-Funktion zur Basis 10 kann hilfreich sein.
    - Die Modulo-Funktion sollte ebenfalls nicht vernachlässigt werden.
    - Lokale Variablen sind bei dieser Art Funktion ungemein wichtig.
- Erstelle eine Funktion “GetTotalDigit”, die eine einzelne Stelle der Summe anzeigt.
    - Hierbei sind die Power- und Modulo-Funktionen zu verwenden.
- Das NumberItem-Widget soll zwei Variablen bekommen: ein Boolean “Active” und einen Integer “Number”. Erstelle ein Text-Element zentriert im Widget und erhöhe die Größe ein wenig. Der Text soll ein Binding erhalten. Folgendes muss eingehalten werden:
    - Wenn das NumberItem-Widget nicht Aktiv ist, soll kein Text angezeigt werden.
    - Ansonsten soll der Text die Number anzeigen.

- Das AutoGetNumberItem-Widget bekommt eine Variable “Index” vom Typ Vector2. Erstelle ein Button-Element und darin ein Text-Element.
    - Das Button-Element soll Aktiv sein, wenn der aktuelle Index noch nicht gesetzt ist.
    - Der Text kann die gleiche Bindung erhalten, wie das NumberItem.
    - Der Button soll beim Klick die Funktion “SetNumber” auf dem GameState aufrufen.
- Das TotalDigitNumberItem-Widget bekommt eine Variable “Digit” vom Typ Integer. Erstelle ein Text-Element mit den gleichen Bedingungen wie im NumberItem.
    - Im Tick-Event soll nun die Number-Variable des eigenen NumberItem auf das Ergebnis der Funktion “GetTotalDigit” des GameState gesetzt werden.
- Lege nun neun AutoGetNumberItem-Widgets im GameWidget an. Gib jedem einen eindeutigen Namen und platziere sie regelmäßig (wie im Beispiel gezeigt)
    - Jedes Widget muss Aktiv sein und einen Index zugeordnet haben.
- Erstelle rechts ein normales NumberItem-Widget.
- Erstelle vier TotalDigitNumberItem-Widgets am unteren Rand des GameWidget.
- Setze die Anchor so, dass die AutoGetNumberItem-Widgets zentriert in der Mitte sind.
- Das NumberItem-Widget soll sich immer rechts ausrichten.
- Die TotalDigitNumberItem-Widgets sollen sich entsprechend am unteren Rand zentriert ausrichten.

Tipps / Hinweise:

- Nutze Vector2D zur Darstellung des Index
- Macros eignen sich besonders für die Umrechnung des 2D-Vektors in ein Array-Index
- Nutze die Node RandomIntRange, um eine Zahl zwischen 1 und 6 zu bekommen.

Erwartetes Ergebnis



Bewertungskriterien

- Sauberkeit der Blueprints
- Konsistente Benennung
- Funktionalität