Dokumentation – Game of Life Projekt

CellState: Gioele, Anton, Richard

Definiert 2 Attribute, *ALIVE* und *DEAD*. Diesen wird jeweils ein Wert hinzugefügt, damit zwischen lebenden und toten Zellen unterschieden werden kann.

Attribute:

- ALIVE: Status einer lebenden Zelle
- DEAD: Status einer abgestorbenen Zelle

Cell: Gioele, Anton, Richard

Die Klasse modelliert die Zellen auf der Benutzeroberfläche, beschreibt also den Bauplan für eine der vielen Zellen.

Attribute:

- X: Spalte, in der die Zelle liegt
- Y: Zeile, in der die Zelle liegt (X und Y machen die Position auf dem Feld, im Grid aus)
- State: Variable vom Typ CellState, gibt Status an, lebendig oder tot
- Next_state: gibt nächsten CellState der Zelle an, wird nach der Anwendung der Regeln gespeichert
- Time_not_changed: Anzahl Durchläufe, die die Zelle sich schon nicht verändert hat in ihrem Status
- Freezed: Bool, gibt an, ob die Zelle sich verändern darf oder nicht (durch Freeze Zauber)

Methoden:

- Determine_next_state:
  - der nächste Stand (ALIVE oder DEAD) wird nach Regeln bestimmt, durch die Anzahl der Nachbarn
  - Anzahl der Durchläufe erhöht, wenn sich der Status der Zellen nicht ändert
- Update_state:
  - Speichert in der Status Variable der Zelle den nächsten, in der vorherigen Funktion bestimmten, Status

Grid:

Die Klasse ist für den Entwurf des Rasters zuständig, des Feldes, auf dem sich die Zellen befinden und ausbreiten.

Attribute:

- Width: Anzahl der Spalten
- Height: Anzahl der Zeilen
- Cell_size: Seitenlänge der quadratischen Zellen
- Cells: das Raster der Zellen, das Feld

- Stats: Array zum Zählen der Anzahl an ALIVE-Zellen und DEAD-Zellen (NewAlive und NewDead später nicht mehr verwendet)

Methoden:

- apply_rle_pattern: Richard
  - Ein vorgefertigtes RLE-Pattern wird auf das Grid angewendet
- parse_rle: Richard
  - Das RLE-Pattern wird in ein 2d-Grid umgewandelt
- Initialize_random: Gioele, Anton, Richard
  - Das Grid wird zufällig mit ALIVE- und DEAD-Zellen befüllt
- Change_cell_state: Gioele, Anton, Richard
  - Der Status der Zelle mit der Position (x,y) im Grid wird getauscht
- Initialize_manually: Gioele, Anton, Richard
  - Das Feld wird nur mit toten Zellen befüllt, sodass der Nutzer frei Zellen auswählen kann
- Reset_field: Gioele, Anton, Richard
  - Alle Zellen auf dem Feld werden zurückgesetzt (auf Status DEAD), der Stand aktualisiert
- Get_neighbours: Gioele, Anton, Richard
  - Findet alle Nachbarzellen einer Zelle und gibt diese zurück
- Update: Gioele, Anton, Richard
  - Bestimmt die Nachbarn für jede Zelle
  - Mit den Nachbarn wird für die Zelle der nächste Stand bestimmt
  - Der Stand (Status) wird schließlich aktualisiert
- Apply_lightning: Gioele, Richard
  - Wendet Lightning-Zauber an (ALIVE und DEAD-Zellen werden vertauscht)
  - Abstand von der Zelle bis zur Mausposition bestimmt, muss kleiner gleich 10 sein
  - Dann Status der Zelle tauschen
- Apply_freeze: Gioele, Richard
  - Wendet Freeze Zauber an
  - Schaut, ob Zelle im Kreis ist mit Radius 10, das Freezed Attribut wird auf True gesetzt
- Apply_unfreeze: Gioele, Richard
  - Alle Zellen werden auf freezed = False gesetzt, sind also nicht mehr gestoppt
- Apply_earthquake: Gioele, Richard
  - Alle ALIVE- und DEAD-ZELLEN vertauscht, nicht nur in einem Kreis wie beim Lightning
- Get_stats: Gioele, Anton, Richard
  - Derzeitige Stats berechnen, wie viele leben, wie viele nicht mehr leben
- Draw: Gioele, Anton, Richard
  - Bringt das Feld auf die GUI, vorher Anpassung des 2D-Arrays an Zoom
  - Berechnung der Farben basierend auf dem Fortschritt der Zelle (ALIVE, DEAD (wie lange DEAD nicht verändert))
- Adjust_Grid: Anton (Zoom-Effekt)

- Expandiert oder verringert die Größe des Feldes bzw. 2D-Arrays basierend auf dem Wert des Zoom Sliders, der eine Änderung der Anzahl an Spalten und Reihen bewirkt

GameOfLife: Gioele, Anton, Richard

Die Klasse kontrolliert den Spielfluss und die anderen Klassen.

Attribute:

- Width: Anzahl der Spalten des Feldes
- Height: Anzahl der Reihen des Feldes
- Cell_size: Größe der quadratischen Zelle
- Grid: Objekt der Grid Klasse

Methoden:

- Initialize:
  - Initialisiert das Feld mit DEAD-Zellen, sodass der Nutzer das Muster bestimmen kann (auf das Feld klicken, vorgefertigte Muster verwenden)
  - Es wird die Funktion der Grid Klasse aufgerufen
- Initialize_automatically:
  - Zufälliges Muster wird erstellt (Funktion der Grid Klasse aufgerufen)
- Next_generation:
  - Ruft die Update Methode aus der Grid Klasse auf
  - Dort wird die nächste Generation bestimmt
- Apply_spell:
  - Ruft den Zauber je nach angegebener Zahl auf

Slider: Anton

Die Slider Klasse modelliert einen selbstgebauten Slider, der auf der GUI positioniert und verwendet werden kann (für die Geschwindigkeit- und Zoom-Features).

Attribute:

- GRAY: graue Farbe für die Linie des Sliders
- BLUE: blaue Farbe für den Kreis auf dem Slider
- X: x-Koordinate des Sliders
- Y: y-Koordinate des Sliders
- Width: Breite des Sliders
- Height: Höhe des Sliders
- Min_value: Mindestwert des Sliders
- Max_value: Maximalwert, den der Slider annehmen kann
- Value: Derzeitiger Wert des Sliders
- Circle_radius: Vorbestimmter Radius des Balls
- Circle_x: Position des Sliders auf der Linie
- Circle_y: y-Koordinate des Kreises (entspricht der der Linie)
- Is_dragging: Bool-Wert, gibt an, ob der Kreis gezogen wird

Methoden:

- Draw:
  - Zeichnet die Linie und den Kreis
- Update:
  - Kontrolliert das Attribut *is_dragging* basierend auf der Mausposition und dem Mausklick
  - Neue Position des Kreises bestimmt (mit Beschränkungen, falls die Maus über die Linie hinaus geht)
  - Berechnet neuen Wert basierend auf der Position des Kreises
- Is_hovering:
  - Schaut, ob die Maus auf dem Slider ist
- Change_value:
  - Ändert den Wert und gleichzeitig die Position des Kreises auf dem Slider
  - Wichtig für Kontrolle der Geschwindigkeit durch Pfeiltasten

SupabasePatterns (Datei): Gioele

Die Datei implementiert die Logik der Datenbank.

Attribute:

- SUPABASE_URL: URL zur Datenbank
- SUPABASE_KEY: Zugriffsschlüssel für die Datenbank
- Supabase: Objekt von der Datenbank

Methoden:

- GetPatterns:
  - Holt Muster von der Datenbank
- addPattern:
  - Lädt Muster in die Datenbank hoch
- deletePattern:
  - Löscht Muster aus der Datenbank

GUI: Anton

Die GUI-Klasse baut die Benutzeroberfläche und enthält die Main loop.

Die Attribute bzw. der Code der GUI-Klasse ist in Form von Kommentaren im Quellcode beschrieben.