Datenbank

Character:

|  |  |
| --- | --- |
| CHARACTERID | string |
| AGE | int |
| BIRTHDATE | timestamp |
| DEATHDATE | timestamp |
| RACE | string |
| BEMERKUNG | string |
| POSITION\_X | float |
| POSITION\_Y | float |
| PROJECT\_ID | string |

Name:

|  |  |
| --- | --- |
| NAMEID | string |
| CHARACTER\_ID | string |
| VALUE | string |
| POSITION | int |

Tags:

|  |  |
| --- | --- |
| TAGID | string |
| CHARACTER\_ID | string |
| VALUE | string |

Relations:

|  |  |
| --- | --- |
| RELATIONID | string |
| CHARACTER1 | string |
| CHARACTER2 | string |
| BEMERKUNG | string |
| BESCHRIFTUNG | string |

Project:

|  |  |
| --- | --- |
| PROJECTID | string |
| PROJECTTITLE | string |

Klassen

|  |
| --- |
| MAP |
| nodeList : LinkedList<Node>  connectionList : LinkedList<Connection>  nodeCount : int  connectionCount : int |
| void CreateNode()  void CreateRelation(Node a, Node b)  void Load()  void Save() |

Save() wird nicht benötigt;

Load ruft der Reihe nach immer wieder CreateNode() auf, bis alle Nodes aus der DB erstellt sind und weist jeder (aufsteigend) eine Nummer zu (id); Dann ruft es CreateRelation() für alle Relations in der Datenbank auf;

Save() ruft der Reihe nach in allen Nodes aus nodeList Save() auf; Danach für die connectionList

|  |
| --- |
| NODE |
| pos\_x : float  pos\_y : float  NODEID : string  id : int |
| void Save() |

NODEID ist die aus der DB; (eindeutige ID)

id ist die ID für die Interaktion zwischen Nodes

Save() speichert alles aus der Node in der DB (falls noch kein Eintrag vorhanden, dann wird neuer erstellt)

|  |
| --- |
| CONNECTION |
| nodeA : Node  nodeB : Node  RELATIONID : string  id : int  beschriftung : string  bemerkung : string |
| void Save() |

Bedienung

* 3 Szenen: Start; Allgemein; Suchresultate
* Jedes Eingabe Popup hat ein Abbrechen und ein Speichern
* Beim Start eine Szene geöffnet, in der die Verfügbaren Projekte aufgelistet werden
* Option für neues Projekt -> Popup mit Aufforderung zur Namenseingabe
* Option, den Namen eines Projekts zu ändern
* Wenn Projekt offen:
* Ribbon am oberen Rand mit
  + New Node (strg + n)
  + New Relation (strg + r)
  + Bearbeiten
  + Suche
  + Zoom
* Ribbon am linken Rand mit einer Liste aller Charaktere
* Wenn Node erstellt wird, kommt Popup mit Eingabeaufforderung für
  + Name (ein Feld, so viele Name wie man will, durch Leerzeichen getrennt)
  + Rasse
  + Geburtsdatum
  + Alter
  + Bemerkung (anders als die anderen mehrere Zeilen)
* Relation erstellen
* Popup am oberen Rand unter dem Ribbon mit der Aufforderung, 2 Nodes anzuklicken
* Verbindung zwischen den zwei Nodes wird gesetzt
* Popup mit Eingabeaufforderung zur Beschriftung und zur Bemerkung
* Bei Klick auf Node oder Connection wird der ausgegraute Bearbeiten-Button aktiv
* Eingabe-Popup wird wieder aufgerufen, aber mit den wesentlichen Werten schon eingetragen
* Klick auf Suche ruft ein Popup auf mit einem Eingabefeld über mehrere Zeilen
* Input erfolgt über „Name : xxxx“ := „where name = xxxx“
* Checkbox für Relation und Character (was durchsucht werden soll; immer nur eines von beidem)
* Hat Abbrechen und OK
* Search-Criteria werden in search.search (oder so ähnlich) gespeichert
* Wird gelöscht, nachdem es ausgelesen wird
* Szene Suchresultate wird geöffnet und der Manager dort ruft search.search auf und lädt nur die gewollten Nodes und deren Relations
* Wenn man Node oder Connection anklickt, erscheint am rechten Rand ein Fenster mit den Daten dazu
* Evtl noch Optionen einführen: Node-Color/Connection-Color (in der RB in dem Table abspeichern)
* Wenn man eine neue Relation erstellt, gibt man den Namen ein
* Klickt dann auf einen Button „CharacterA“
* Das Fenster wird außerhalb des Kamerabereiches geschoben
* Sobald der User auf eine Node klickt, wird diese selected und ihr Name in dem Fenster angezeigt
* NodeB ´´
* Dann Bemerkung
* Dann Accept