Ayaz Khudhur Alexey Khokhlov

Zyklus 5

Inhalt

Freie Aufgabe (b.z. 10p)

Beschreibung der Aufgabe

- Erstellen eines Grafischen Inventars
- Gegenstände sollen verwendbar sein (durch Maus Klick)
- Das grafische Inventar soll so viele Slots rendern, wie das Inventar Platz hat (dynamische Slot Anzahl)
- Das grafische Inventar soll nicht nur für den Helden „verwendbar“ sein, sondern auch für Schatzkisten/Shop
- Das Spiel wird pausiert, sobald sich das Inventar öffnet
- Wenn Gegenstände ausgerüstet sind, z.B. ein Schwert und im Inventar ein anderes Schwert gewählt wird sollen diese „Tauschen“, das momentan ausgerüstete Schwert soll im Inventar landen und das neu ausgerüstete Schwert aus dem Inventar „verschwinden“

Beschreibung der Lösung

Für das Inventar orientieren wir uns an das „Dialog System“, in unserem Projekt existiert bereits ein Dialog System mit dem Geist. Das grafische Inventar benötigt für den Aufruf die „Aufrufer Klasse (this)“, zusätzlich muss zwischen Schatzkisten/Shop und Helden Inventar unterschieden werden. Für jeden „Inventarplatz“, der Entity wird ein Slot gerendert (positionsversetzt), ist ein Item auf dem Slot vorhanden wird das Item Inventar Icon gerendert, ansonsten eine leere „Kachel“. Sobald das Inventar aufgerufen wird, wird das Spiel mit den Methoden in der Game Klasse pausiert. Die Unterscheidung ob das Entity der Held ist oder etwas anderes „loot-/handelbares“ ist wird wichtig sein, um Items zu tauschen (Waffenhand Item zurück zum Inventar), Schatzkisten und Shop haben diese Möglichkeit nicht, stattdessen erhalten diese Entitys eine Methode, um ein Item aus Ihrem Inventar in das Inventar des Helden zu schieben.

Methoden und Techniken

Der Code wird mit JavaDoc ggf. mit normalen Kommentaren (private) dokumentiert. Logger werden an sinnvollen Stellen gesetzt (Öffnen/schließen des Inventars, vorhandene Items, Verwendung eines Items, entfernen eines Items)

Das Inventar wird auf die Methoden und Techniken des Dialog Systems aufbauen.

Ansatz der Modellierung

Der Grund Aufbau wird auf dem bereits Implementierten Dialog System stammen. Das Inventar wird zwei Parameter benötigen „this“ und ein „boolean“ mit dem der Held von anderen Entitys unterschieden wird. Das Grafische Inventar soll sich mit der Taste „I“ öffnen lassen (ggf. auch „B“). Für das Zeichnen und die klick Methoden werden die GDX Bibliothek Methoden verwendet. Der boolean Parameter, der von der Aufrufenden Klasse gesetzt wird dient als Sicherheit, dass Methoden die eine Entity nicht haben sollte auch nicht ausgeführt werden können, bei anderen Entitys außer dem Helden wird das Inventar auf der linken Seite angezeigt, beim Helden rechts.

Das Inventar wird mit der update() Methode der jeweiligen Entity aufgerufen, Kollidiert ein Held mit einer Entity die über ein Inventar verfügt wird das Inventar der Entity und des Helden angezeigt, wenn auf die Taste „E“ gedrückt wird.

Die Methode checkItem(int) wird aufgerufen, wenn ein Item benutzt wird, hier findet z.B. die Fallunterscheidung statt ob ein Item „verbraucht“ wird, eine Waffe ist und mit der aktuellen Waffe, die ausgerüstet ist tauschen muss oder das Item in das Helden Inventar verschoben wird.

Die Methode useItem(Int) wird aufgerufen, wenn ein „Verbrauchs“ Item benutzt wird

Die Methode weaponItem(int) wechselt die Waffen aus vom ausgerüsteten Waffenslot zurück zum Inventar und die Waffe, die sich im Inventar befindet in den Waffenslot.

Die Methode moveItem(Int) überträgt ein Item von einer beliebigen Entity in das Helden Inventar. (Zurücklegen haben wir nicht vorgesehen, wenn Zeit dafür bleibt, kommt das auch)

Die Methode renderInventory() zeigt das Grafische Inventar an.

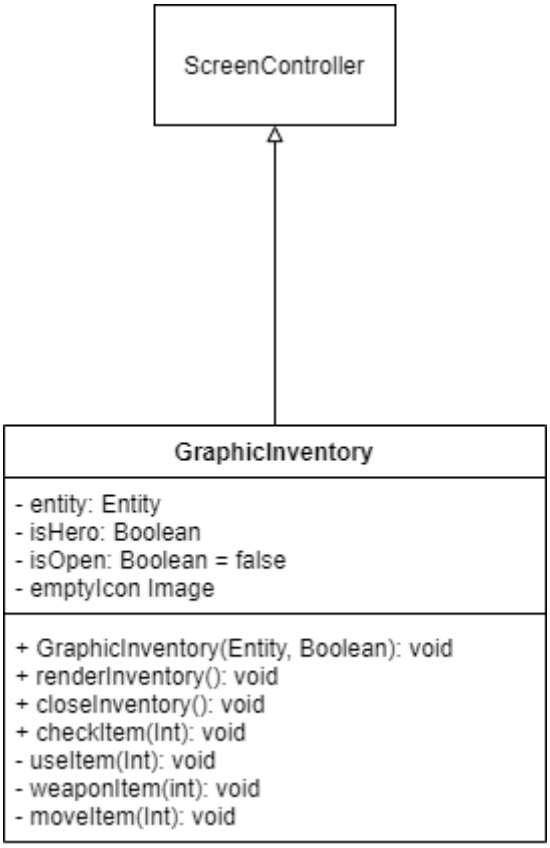
Die Methode closeInventory() „schließt“ das Grafische Inventar.

Grundeigenschaften

- Inventar:
  Bekommt eine Entity übergeben und ein boolean true/false

- Betreffende Entitys:
  Rufen in der update() Methode renderInventory() auf, wenn die Taste gedrückt wird

UML



Beschreibung

Graphisches Inventar:

- Entity entity // Das Entity, das das Graphische Inventar haben soll
- Boolean isHero // Gibt an ob das Entity der Held ist oder nicht (Vermeidung von typeof)
- Boolean isOpen // Anzeigestatus des Inventars
- Image emptyIcon // Leere Inventar Kachel, wenn ein Inventarplatz frei ist

Held:

- Item weaponHand // Ausgerüstetes Item in der Waffen Hand
- GraphicInventory graphicInventory // Initialisierung des Grafischen Inventars

Freie Aufgabe (b.z. 10p)

Beschreibung der Aufgabe

Folgende Elemente/Skill werden auf dem Benutzeroberfläche (UI) sichtbar.

- Skills
- Fire
    - Lightning
    - MindControl
    - Transform
- Cooldown bis zu der nächsten Aktivierung
- Hero Level
- Hero HP
- Hero XP
- Hero Mana
- Dungeon Level
- Sowie die Tastenbefehle für Skills

Beschreibung der Lösung

Für die Skalierung der Images & Position werden wir uns an dem GameOver-Screen orientieren. Eine neue Klasse wird hinzugefügt und jeden Stichpunkt bekommt eine eigene Methode, um im Konstruktor den Code sauber zu halten. Für Skills wird die Klasse ScreenImage benutzt und für restlichen Attributen ScreenText verwendet. Position: Skill werden mittig angezeigt, die Mana und XP sind Linksbündig und HP & Level Rechtsbündig

Eigen Assets werden verwendet!

 Skill Auswahl Background

 FireballSkill

 MindControlSkill

 LightningSkill

 TransformSkill

Beispiel für den Hero Level

Methoden und Techniken

Der Code wird mit JavaDoc ggf. mit normalen Kommentaren dokumentiert. Logger werden an sinnvollen Stellen gesetzt.

Der UI_HUD wird auf die Methoden und Techniken des GameOver-Screens aufbauen.
Ansatz der Modellierung
Wir erstellen eine Klasse namens UI_HUD, die über folgende Eigenschaften und Methoden verfügt:
6 finale Attribute vom Typ String.
Methoden zum Anzeigen von Bildern/Texten auf der Benutzeroberfläche (UI).
Der Konstruktor ist so gestaltet, dass der Code übersichtlich bleibt.
Die Klasse UI_HUD erbt von ScreenController, um auf die Methode "add()" zugreifen zu können. Dadurch können wir dem Controller unsere Elemente/Bilder und deren Positionen übergeben. Mit der Anweisung "(Actor s)s.setVisible(true);" werden letztendlich die UI-Elemente auf dem Bildschirm angezeigt und für den Spieler sichtbar gemacht.

UML