Game Developement

Am Beispiel von Frogger nach der Lernplattform tigerjython.ch

Detailliertes Szenario aufschreiben

1. Frosch muss über eine stark befahrene Strasse zu seinem Teich (Bewegung mit Cursortasten).
2. Die Fahrzeuge bewegen sich von Links nach Rechts und von Rechts nach links (4 Fahrbahnen: 2 mit gegeneinander fahrenden Bussen und Lastwagen und 2 mit gegeneinander fahrenden Oldtimer-Autos).
3. Bei einer Kollision mit einem Fahrzeug verliert der Frosch ein Leben (Verrechnung von Punkten etc.).

Das Spielefenster

- *from gamegrid import **
- Klasse *GameGrid* aus der Gamelibrary JGameGrid modellieren:
  makeGameGrid(800, 600, 1, None, "sprites/lane.gif", False)
  makeGameGrid(#x, #y, #Pixel, Linienfarbe, Hintergrundbild, Navigationsbalken)
- *show()*

Bewegung der Fahrzeuge

- Fahrzeuge werden als Instanzen der Klasse *Car* modelliert
- In der Methode *act()* wird die Bewegung der Fahrzeuge programmiert
- In der Funktion *initCars()* werden die 20 Car-Objekte mit entsprechender Stelle und Blickrichtung erzeugt.
- *setSimulationPeriod(50)*
  50ms = Spielszene wird 20 Mal pro Sekunde gerendert
- *initCars()*
  show()
  doRun()

Frosch mit Cursortasten bewegen

- Die Klasse *Frog* als Unterklasse von *Actor* initialisieren.
- Der Frosch benötigt keine weiteren Methoden, da er mit den Tastaturevents gesteuert wird.
- Den callback *onKeyRepeated* definieren.
- *onKeyRepeated* in *makeGameGrid()* mit dem benannten Parameter *keyRepeated* registrieren.
- *addActor(frog, Location(400, 560), 90)*

Kollisionsevents

- Neue Methode bei der Erzeugung eines Fahrzeugs (*initCars()*):
  frog.addCollisionActor(car)
  Diese Methode löst bei jeder Kollision mit einem Fahrzeug einen Event aus (die Methode *collide()* aus der Klasse *Actor*).
- Die Methode *collide()* wird in der Klasse *Frog* überschrieben.
- Die Methode *self.setCollisionCircle(Point(0, -10), 5)* definiert den Kollisionsbereich (*setCollisionCircle(centerPoint, radius)*).

Game-Supervision

- Erfolg:
  Neue Methode in der Klasse *Frog* definieren (*act(self)*).
  Darin eine global variable definieren (*nbSuccess*) die beim Erreichen der anderen Strassenseite eine Tonfolge abspielt und den Frosch an die Urpsrungsposition zurücksetzt.
- Misserfolg:
  In der Methode *collide(self, actor1, actor2)* wird eine global variable definiert (nbHit) die jeder Kollision eine Tonfolge abspielt und den Frosch an die Urpsrungsposition zurücksetzt.
- Schleife, die das Spiel überwacht (*while not disposed()*)

Game-Projekt: Ihr Auftrag

- Sie entwickeln in einer Arbeitsgruppe von maximal 3 Personen ein eigenes Computer-Spiel mit Tigerjython und der libary gamegrid.
- Ihr Projekt geben sie zusammen mit folgenden Beilagen ab:
  -Kommentierter Code
  -reflektierter Prozess (in Textform von maximal 2 Seiten) mit Fachbegriffen aus der OOP (evtl. Klassendiagramme) und einem Projektplan.
- Sie präsentieren ihr Spiel innerhalb von maximal 10 Minuten. Dabei muss jeder Gruppenteilnehmer ein Vortragender sein. Das Publikum hat nach der Präsentation die Möglichkeit Fragen zu stellen, die die Gruppe kompetent beantworten soll. Dabei soll jeder Gruppenteilnehmer die gestellten Fragen beantworten können.

Game-Projekt: Notenkriterien

- Spiel: Komplexität, Authentizität, Kreativität
- Implementierung: OOP-Modellierung, algorithmisches Denken und Python-Konventionen des kommentierten Programmes.
- Kommentierung: Authentizität und reflektorische Tiefe
- Handout: Detailliertes Spieleszenario und Erklärung der einzelnen Entwicklungsschritte.
- Präsentation: Klarheit der Darstellung, der Sprache und der Antworten auf gestellte Fragen.
- Gewichtung Produkt/Präsentation: Das Produkt wird als Gruppenprodukt gewertet. Durch die Präsentation kann diese individuell verändert werden, wodurch die Endnote entsteht.

Game-Projekt: Termine

- 14. September um 23.59 muss das Projekt auf github hochgeladen oder per Mail an mich gesendet sein (benjamin.senn1@edubs.ch).
- 17. September alle Präsentationen
- 19. September Test OOP-Theorie

Inspiration unter: jython.ch