Brändi Dog 2.0 Technologien

Anforderungen

Das Coding der Spielregeln und des Gamestates ist für unser Brettspiel überschaubar und geschah direkt (und „von Hand“) via IDE, ohne dass hierfür zusätzliche Technologien hinzugezogen werden müssten.

Schwierigkeiten sahen wir bei Erstellung und Darstellung der grafischen Oberfläche. Dazu mussten Grafikprogramme zur Erstellung der Assets verwendet werden und es musste eine Lösung für die Generierung des GUI gefunden werden.

Fürs Testing der Methoden und Klassen wird nach Aufgabenstellung Junit 4, eine externe Library verwendet.

Ton ist in diesem Projekt zwar optional, dennoch wollten wir uns daran versuchen. Es wurde eine Software zur Erstellung und Bearbeitung der Soundeffekte benötigt. Ausserdem war eine Lösung gefragt, um unser Programm wiedergabefähig zu machen. Libraries waren eine Möglichkeit, wir entschieden uns jedoch für eine Java-interne Lösung.

Für Entwicklung und Versionskontrolle wurden uns bereits Lösungen von der Universität zur Verfügung gestellt.

Technologien

Libraries

JUnit

Aus der Aufgabenstellung geht bereits hervor, dass wir JUnit verwenden werden. JUnit ist ein Framework, mit dem Klassen und Methoden praktisch getestet werden können (Unit Tests).

JavaFX

Obwohl JavaFX mittlerweile zur standard SE von Java hinzugehört, soll es hier kurz erwähnt werden. Sowohl das Chat-GUI wie auch das Spiel-GUI wurden damit erstellt.

IDE: Eclipse und Intellij

Statt lange darüber zu streiten, auf welche eine und einzige und korrekte IDE wir uns beschränken, überlassen wir das jedem Mitglied selber. Solange die Konventionen eingehalten werden, darf auch Notepad verwendet werden (zum Glück war niemand von uns wahnsinnig genug, das zu tun). Eclipse und Intellij haben sich durchgesetzt. Beide bieten uns Git- und auch Gradle-Integrierung an und nehmen uns somit etwas Arbeit ab.

Git, Gradle und Continuous Integration

Von der Universität wird uns ein Git-Repository zur Verfügung gestellt, mit welchem wir die Entwicklung unserer Projekts aufzeichnen (Versionskontrolle). Es ist auch eine Anforderung aus der Aufgabenstellung.

Gradle ist auch bereits ins unser Projekt integriert. Das Build-Tool nimmt uns die Arbeit ab, für jeden neuen Build manuell einen .jar-Container sowie die Dokumentation selbst zu erstellen.

Um das Building noch mehr zu automatisieren wollen wir uns auch um Continuous Integration kümmern. Mit CI wird das Testing und Building mit jeder Änderung am Code automatisiert.

Weitere Software

Grafiksoftware Gimp und Photoshop

Die Spielgrafik wird mit JavaFX angezeigt, doch erstellt wurde sie mit Gimp. Dabei handelt es sich um eine Bildbearbeitungssoftware, vergleichbar mit Photoshop. Ausserdem wurde Photoshop verwendet, um die Logos von Spiel und Entwickler zu erstellen. Ähnlich wie schon bei den IDEs hatten sich privat schon vor Projektbeginn verschiedene Softwaren bei verschiedenen Personen durchgesetzt. Und wie bei den IDEs ist uns das Ergebnis wichtig, nicht das verwendete Tool.

Audosoftware Logic Pro

Für die Implementation von Soundeffekten, mussten diese zuerst erstellt werden. Dies geschah mit der Tonbearbeitungssoftware Logic Pro. Logic ist in etwa wie Photoshop, nur halt für Tonbearbeitung statt Bildbearbeitung.