PlayN

(http://playn.io/)

Überblick

- PlanN ist eine Java-Bibliothek für die Programmierung von Spielen
- es können Spiele für Windows / Mac / Linux Desktop Java, iOS-Geräte, Android-Geräte und HTML5-Browser entwickelt werden (mehr unter **Desktop Java**)
- kostenlos, Open Source
- Lizenz: unter der Apache 2.0 Lizenz veröffentlicht
- am einfachsten zu verwenden mit Maven (ein auf Java basierendes Build-Management-Tool)

Features

- Unterstützung für die Bereitstellung von Spielen auf Desktop, Browsern und Mobilgeräten
- Bereitstellung zahlreicher Zeichen-APIs, PlayN ist in erster Linie eine 2D-Bibliothek (mehr unter **Mehr zur Grafik**)
    - 2D (CPU) Canvas Zeichen API mit Text Rendering
    - zahlreiche auf OpenGL basierende APIs
- Musik- und Soundeffektwiedergabe
- Maus-, Touch- und Tastatureingabe
- Die TriplePlay Bibliothek baut auf PlayN auf und bietet ein UI-Toolkit und verschiedenste Unterstützungen für Animationen
- PlayN bringt keinen eigenen Editor mit, es ist aber kompatibel mit Eclipse, Netbeans, IntelliJ und den meisten anderen IDEs, die für die Java-Endwicklung verwendet werden
- ein eigenes Testframework oder eine Assetbibliothek gibt es bei PlayN nicht

Community & Tutorials & Dokumentation

- es gibt vor allem 2 Plattformen, welche auch auf der offiziellen PlayN Website beworben werden:
    - Stackoverflow: 2 Fragen innerhalb der letzten 5 Jahre
    - PlayN Google Group: 7 Fragen/ Diskussionen innerhalb der letzten 12 Monate

  $\rightarrow$ kleine und nicht sonderlich aktive Community

- keine große Auswahl an Tutorials, nur ein einziges (auf der offiziellen PlayN Seite) verfügbar (http://playn.io/docs/reversi-tutorial.html)
- im Allgemeinen keine weitere Dokumentation außer den offiziell beworbenen (auf der PlayN Seite und teilweise im GitHub)

Desktop Java

- das Desktop-Java-Backend verwendet eine Standart- Java-VM und LWJGL (Lightweight Java Game Library) um unter den unterstützen Desktop-Betriebssystemen (Windows, Max OS und Linux) auf OpenGL zuzugreifen
- mit diesem Backend kann man alle hoch entwickelten Tools, die im Java-Ökosystem verfügbar sind, nutzen

Mehr zur Grafik (http://playn.io/docs/overview.html#graphics)

- Klasse GL20
  - PlayN baut alle Grafikdienste auf OpenGL auf
  - OpenGL = Spezifikation einer plattform- und programmiersprachenübergreifenden API zur Entwicklung von 2D- und 3D-Computergrafikanwendungen (als Zustandsautomat entworfen, um den Code zu vereinfachen und unnötige Reorganisationen der Grafikpipeline zu vermeiden)
  - die PlayN Klasse GL20 macht OpenGL ES 2.0 API direkt verfügbar
- Klasse Disposable
  - kann verwendet werden, um GPU Ressourcen sofort frei zu geben, sobald sie nicht mehr gebraucht werden
- Klasse Rendertarget
  - rendern von OpenGL in eine Texture für die spätere Wiederverwendung $\rightarrow$ Vereinfachung des Codes
- Klasse QuadBatch
  - Klasse, um Quads auf den Bildschirm zu zeichnen
  - Quad = Polygon (Form) welche vier Seiten und vier Punkte hat
  - jeder Quad wird mittels sechs Floats definiert, so dass mehrere willkürlich transformierte Quads in einem einigen Batch (Stapel) gezeichnet werden können
- Klasse Surface

    - ermöglicht benutzerfreundlicheres Zeichnen als QuadBatch
    - wird am Ende trotzdem als QuadBatch gerendert
- Klasse Canvas
    - zum Zeichnen von Canvas-Elementen (vielfältiger als Quads, Objekte können wie bei OpenGL in einem Stack abgelegt werden)
    - Unterstützung zum Text Rendering
- 2D Scene Graph

Einschätzung bezüglich der Eignung für unsere Aufgabe

- **Pro**
    - Server-plattformübergreifen, kostenlos, Open Source
    - Unterstützung für die Bereitstellung von Spielen als Desktop Java Anwendung
    - für und in Java geschrieben
    - in erster Linie eine 2D-Bibliothek
        $\rightarrow$ völlig ausreichend für unsere Aufgabe
    - OpenGL (worauf PlayN alle Grafikdienste aufbaut) ist direkt über die Klasse GL20 verfügbar
    - zahlreiche Zeichen-APIs
    - alle Tools, die im Java-Ökosystem verfügbar sind, können genutzt werden

- **Kontra**
    - kleine und nicht sonderlich aktive Community
    - kaum Tutorials verfügbar (nur ein einziges auf der offiziellen PlayN Seite)
    - im Allgemeinen keine weitere Dokumentation außer den offiziell beworbenen (auf der PlayN Seite und teilweise im GitHub)
    - keine eigene Assetbibliothek
    - kein eigenes Testframework

- **Fazit**
    - ich würde mich tendenziell gegen die Nutzung von PlayN für unsere Aufgabe aussprechen, vor allem wegen der kleinen, wenig aktiven Community und da es neben der offiziellen

Dokumentation und dem Tutorial keine weiteren Dokumentationen oder Tutorials gibt

- außerdem lässt ein Blick in eine Plattform zum Austausch bezüglich PlayN vermuten, dass die Software noch einige bugs hat (https://groups.google.com/g/playn)
  → sollten wir im Verlauf unseres Projekts auf einen bug stoßen, den wir beheben müssen, wäre das einen zusätzlichen, unnötigen Aufwand
- daher denke ich, dass PlayN einfach noch nicht optimal ausgereift ist und eine zu kleine Community hat, weswegen es sicher bessere geeignete Software für unsere Zwecke gibt