Liti Engine

ist eine 2D game libary für Java (basierend aus Java 8)
gerade in Beta
für Windows Linux MacOS → nicht für Android
„[...]simple to pick up, yet powerful enough for creating massive worlds.“
sehr gute Website für Anleitungen (https://litiengine.com/)

2d Physics (`Game.physics()`)

Kollisionserkennung/Verarbeitung
Kräfte auf Entitäten anwenden
Entitäten mit Beschleunigung unter Berücksichtigung von Reibung und Gewicht bewegen

2d Render (`Game.graphics()`)

Texte, Formen, Bilder und Objekte an Position in Umgebung in Bezug auf Position und Zoom der Kamera rendern
Position muss innerhalb des Koordinatenraums der aktuellen Umgebung liegen
übersetzt die Koordinaten in eine Position auf dem Bildschirm

benutzt statische render implementation um auf dem Graphics2D-Objekt zu zeichnen
Codebeispiel:

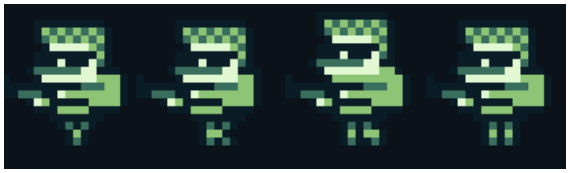
```
// render "my text" at the environment location (50/100)
Game.graphics().renderText(g, "my text", 50, 100);

// render "my text" at the location of an entity
Game.graphics().renderText(g, "my text", myEntity.getX(), myEntity.getY());

// render a rectangle (50x50 px) at the environment location (50/100)
Rectangle2D rect = new Rectangle2D.Double(50, 100, 50, 50);
Game.graphics().renderShape(g, rect);
```

Bilder, Formen und Strings kann man Rendern

Rendern mit Spritesheets (AnimationController):

Entität mit passenden Sprite für den aktuellen Zustand zeichnen durch zugewiesenen Animationcontroller
LitiEngine benutzt single-purpose spritesheets (für jede Animation ein eigenes)
Beispiele:

Animation für Entity „Player“ (vererbt von der default Entity „Creature“)



Sprite mit Namen:
Player-walk-left.png

SpritePrefix-State-Direction.png

`Creature.getSpritePrefix()` setzt SpritePrefix bzw ist direkt im Konstruktor der Entity spezifiziert
es gibt 3 Standard-Zustände (State): idle, walk, dead
Direction kann beliebig einen Wert des Direction-Enums annehmen

wenn die „Direction“ „left“ oder „right“ angelegt ist wird diese für den jeweils anderen gespiegelt. Man muss also nicht beide festlegen bei gleicher Animation

der „CreatureAnimationController“ sucht auch nach Sprites ohne „direction“

rendert auf einer „Java AWT Canvas“ – Komponente, es ist kein explizites OpenGl involviert

2d Sound (`Game.audio()`)

Sound kann an 2D – Koordinaten gespielt werden oder direkt abhängig von Entitäten
es wird .wav, .mp3 und .ogg unterstützt
Codebeispiel:

```
Sound mySound = Resources.sounds().get("my-sound.ogg");

// play the sound
Game.audio().playSound(mySound);

// play the sound at environment location (50/50)
Game.audio().playSound(mySound, 50, 50);

// play the sound at location of an entity
Game.audio().playSound(mySound, myEntity);

// play background music
Game.audio().playMusic(Resources.sounds().get("my-music.ogg"));

// use the SoundPlayback to react to events
ISoundPlayback playback = Game.audio().playSound(mySound);
playback.addSoundPlaybackListener(new SoundPlaybackListener() {

	@Override
	public void finished(SoundEvent event) {
	}
	@Override
	public void cancelled(SoundEvent event) {
	}
});
```

Sound wird je nach Quellen- und Zuhörerposition angepasst

Basic Game Infrastructure / API (https://litiengine.com/api/)

alle wichtigen Teile für das Spiel direkt über Game – Klasse statisch zugreifbar
Code zum Initialisieren und Starten:

```
public static void main(String[] args) {
	Game.init(args);
	Game.start();
}
```

Code für Eventlistener:

```
Game.addGameListener(new GameListener() {
	@Override
	public void initialized(String... args) {
	// do sth when game is initialized
	}
	@Override
	public void started() {
	// do sth when game started
	}

	@Override
	public void terminated() {
	// do sth when game terminated
	}
});
```

andere Funktionen von der Game-Klasse:

Game

config()
info() z.B. Game.info().getVersion()
metrics()
time()
loop() Die Spiellogik ist von der Framerate abgekoppelt und
renderloop() läuft in einem seperaten Loop
inputloop
world()
window()
screens()

Utility Editor

utiLITI:

Projektmanagement- und Kartenerstellungswerkzeug
LitiEngine kann auch ohne betrieben werden

Features:

erstellt automatisch eine Spielressourcendatei mit allen nötigen Assets
importieren von Maps, Tilesets, Spritesheets, benutzerdefinierte Emitter und Entity Blueprints
Assets werden auch in die xml-Datei aufgenommen die man kodieren kann (um Plagiate zu vermeiden)
in utiLITI erstellte Props, Creatures, Lights, Triggers, SpawnPoints, CollisionBoxes, Areas, StaticShadows und Emitter können sofort in LitiEngine gerendert werden
man kann hier Karteneigenschaften festlegen (Name, Schatten, Farben, Beschreibung, usw.)

Installer: https://litiengine.com/download/

Tile Map (TMX)

.tmx tile maps werden genutzt
empfohlene pixel – painting – Programme (https://www.slant.co/topics/1547/~best-pixel-art-sprite-editors)
Tiled wird dafür empfohlen (https://www.mapeditor.org/download.html), der nicht in utiLITI enthalten ist, da utiLITI keine externen Bibliotheken nutzt

Community

https://forum.litiengine.com/ (Offizielles Forum der LitiEngine)
letzte Aktivität 2 Wochen her
https://discord.com/invite/bWfP4E9 (Offizieller Discord)
https://github.com/gurkenlabs/litiengine (GitHub)

Spiele

Pumpkin Keeper (2020) (https://litiengine.com/pumpkin-keeper/)
Servus Bonus (2019) (https://litiengine.com/servus-bonus/)
Goin – Behave or get lost! (2018) (https://litiengine.com/goin-behave-or-get-lost/)
Naughty Elves (2017) (https://litiengine.com/naughty-elves/)
Naughty Gnomes (2017) (https://litiengine.com/naughty-gnomes/)
DR.LEPUS – The Last Rabbit on Earth (2016) (https://litiengine.com/dr-lepus-the-last-rabbit-on-earth/)
LITI – Stoneage Brawl (2014) (https://litiengine.com/liti-stoneage-brawl/)

Alles Spiele im „Pixelstyle“. Einfache Bewegungs- und Interaktionsmechaniken Mecha Wars ist durch das Rundenbasierte Spielprinzip definitiv hier einfach unterzubringen

Fazit:

Die LitiEngine ist durch utiLITI ein einfaches Tool um verschiedene oberflächen-basierte Spielfelder zu erstellen und zu verwalten. Da Mecha Wars ein Runden-basiertes Strategie-Brettspiel ist, lässt es sich durch diese Engine besonders leicht umsetzen. Dies kennzeichnet auch die Möglichkeit mit utiLITI jedes Objekt von dort aus schnell und modular zu rendern. Auch eine Leichtigkeit stellt das Einbinden von Audio dar.

Quelle:

https://litiengine.com