Übungsblatt 6 – EIPNF WS 16/17

Besprechung des Übungsblatts erfolgt am 09.12. & 12.12.

Aufgabe 1:

Bibliotheken lassen sich einfach in Processing einbinden und ermöglichen es uns Funktionalität, die andere Nutzer entwickelt haben in unseren Projekten zu integrieren.

Wir wollen nun unseren Keyboard-Visual Sketch um Sound erweitern. Bei jedem Tastendruck soll nicht nur eine Animation abgespielt werden, nun soll auch ein kurzer Sound ertönen.

1. Lade die Library „Sound“ herunter:
    1. Klicke auf den Menüpunkt „Sketch“
    2. Wähle im Unterpunkt „Libraray importieren …“ den Punkt „Library hinzufügen …“
    3. Gebe im Suchfeld „Sound“ ein und wähle „Sound“ von dem Autor „The Processing Foundation“ an
    4. Klicke auf „install“ und warte bis der Prozess abgeschlossen ist
2. Importiere die zuvor heruntergeladene Library in unseren schon bestehenden Sketch:
    1. Klicke auf den Menüpunkt „Sketch“
    2. Wähle im Unterpunkt „Libraray importieren …“ den Punkt „Sound“
3. Jede Animation soll einen eigenen Sound abspielen. Lese den Eintrag zum Thema `SoundFile` hier nach: https://processing.org/reference/libraries/sound/
4. Füge der Klasse Animation ein Feld `sound` des Typs `SoundFile` hinzu.
5. Diesem Feld wird im Konstruktor ein als Parameter übergebenes `SoundFile` zugewiesen.
6. Füge in der Methode `start()` der Klasse `Animation` den Befehl hinzu, der die Wiedergabe des `SoundFile` startet (natürlich nur wenn die passende Taste gedrückt wurde).
7. Passe die Konstruktoren der Kindklassen an.
8. Erzeuge im Konstruktoraufruf ein SoundFile, dass als Parameter übergeben wird.

```
// import der Library „Sound“

Animation a,b;

void setup () {
  size(600,400);
  // initialisiere a und b mit je einem Soundfile als Übergabeparameter
  a = new ColorCircle (300,200,'2');
  b = new RotatingRect (300,200,'1');
}

void draw () {
  background(0);
  a.animate();
  b.animate();
}

void keyPressed () {
  a.start(key);
  b.start(key);
}

void keyReleased () {
  a.stop(key);
  b.stop(key);
}



// NEUER TAB ------------------

interface Triggerable {
  void start(char keyInput);
  void stop(char keyInput);
}

interface Animateable {
  void animate();
}



// NEUER TAB ------------------

class Animation implements Animateable, Triggerable {
  char triggerKey;
  int animationCounter;
  // Deklariere ein Feld des Typs SoundFile

  // ergänze einen Übergabeparameter vom Typ SoundFile
  public Animation (char triggerKey) {
    this.triggerKey = triggerKey;
    animationCounter = 0;
    // weise das übergebene SoundFile dem passenden Feld zu
  }

  // ergänze den Aufruf des Wiedergabebefehls auf dem SoundFile
  void start (char keyInput) {
    if (keyInput == triggerKey && animationCounter == 0) {
      animationCounter = 1;
    }
  }

  void stop (char keyInput) {
    if (keyInput == triggerKey) {
      animationCounter = 0;
    }
  }

  void animate () {}
}
```

```
// NEUER TAB ------------------

class ColorCircle extends Animation {
  int x,y;

  // Passe Übergabeparamter an
  public ColorCircle (int x, int y, char triggerKey) {
    // passe Konstruktoraufruf an
    super(triggerKey);
    this.x = x;
    this.y = y;
  }

  void animate () {
    if (animationCounter>0) {
      animationCounter++;

      fill(random(255),random(255),random(255));
      int r = animationCounter%100;
      ellipse(x,y,r,r);
    }
  }
}



// NEUER TAB ------------------

class RotatingRect extends Animation {
  int x,y;

  // Passe Übergabeparamter an
  public RotatingRect (int x, int y, char triggerKey) {
    // passe Konstruktoraufruf an
    super(triggerKey);
    this.x = x;
    this.y = y;
  }

  void animate () {
    if (animationCounter>0) {
      animationCounter++;

      pushMatrix();
      translate(x,y);
      rotate(2*PI*animationCounter*0.3);
      fill(255,0);
      stroke(255);
      rect(-100,-100,200,200);
      popMatrix();
    }
  }
}
```