# Dokumentation zum Projekt „MemorizeApp“

Spielprogrammierung mit dem Kivy-Framework

## Teammitglieder:

Jenny Pilz-Rosenthal

Natalia Prost

## Rahmenbedingungen:

* Projektstart: 13. Juni 2015
* Projektdauer: 3 Wochen
* Abgabe: 5. Juli 2015

## Gruppenarbeit:

* Versionsmanagement: Git-Repository
* Code Conventions: z. B. eineVariable, EINE\_KONSTANTE, eine\_methode
* Pair Programming, nach Möglichkeit jeden Abend:
  + Klassenstruktur, notwendige Methoden besprechen
  + Probleme und Bugs besprechen
  + Anlegen von TODOs
* zwischen dem gemeinsamen Programmieren
  + selbstständige Erarbeitung der notwendigen Kivy-Grundlagen
  + TODOs implementieren und neue erstellen

## Was ist Kivy?

Bei Kivy handelt es sich um ein Open-Source Python Framework. Es wurde erstmals im Februar 2011 veröffentlicht und liegt aktuell (Juni 2015) in der Version 1.9.0 vor.

Aufgrund seiner Python-Basis ist auch Kivy selbst plattformunabhängig. Es ermöglicht die schnelle Entwicklung von plattformübergreifenden Apps und ist neben Linux, Windows und Mac OS X auch auf mobilen Plattformen wie Android und iOS lauffähig. Dabei kann der gleiche Code für alle unterstützten Plattformen genutzt werden.

Neben der Möglichkeit von innovativen Benutzerschnittstellen (Natural User Interfaces, multi-touch Apps), zeichnet es sich durch schnelle Entwicklungsprozesse aus, ... („rapid development“)

*Einfache Kivy-App:*

|  |  |
| --- | --- |
| from kivy.app import App  from kivy.uix.button import Button  class TestApp(App):  def build(self):  return Button(text='Hello World')  TestApp().run() | esult of the example |

## Spielidee:

*Ziel des vorliegenden Projekts war ein 2D-Spiel zur Verbesserung der Merkfähigkeit mit Hilfe einfacher Symbole in verschiedenen Formen und Farben. Es ist als Single Player-Spiel konzipiert und soll auch im mobilen Benutzerkontext durch ein einfaches Bedienkonzept überzeugen.* *Die initiale Spielidee umfasste folgende Punkte:*

* Spieler prägt sich angezeigte Symbole ein
* anschließend werden die Symbole (teilweise) ausgeblendet
* Spieler muss die Symbole nacheinander wieder einfügen
* dazu steht eine Auswahl vom Symbolen zur Auswahl
* die Auswahl des Spielers wird überprüft und bepunktet
* mit wachsendem Erfolg wird es schwieriger, z. B. mehr Symbole

## Anforderungen:

**MUSS-Kriterien:**

* Das Spiel muss eine bestimmte Anzahl von verschiedenen Symbolen anzeigen.
* Das Spiel muss diese Symbolfolge zufällig erstellen.
* Die verwendeten Symbole müssen einfachen grafischen Formen entsprechen, die dynamisch generiert werden. (keine Bilddateien)
* Das Spiel muss eine bestimmte Anzahl der Symbole grafisch „verdecken“. (ähnlich Karten umdrehen bei Memory-Spielen, aber ohne Animation)
* Der Spieler muss aus einer anderen zufälligen Symbolfolge das gesuchte Symbol auswählen können.
* Der Spieler muss visuelles Feedback erhalten, ob seine Wahl richtig oder falsch war.
* Jede richtige Auswahl muss zu einer Erhöhung des Punktestands führen.
* Das Spiel muss nach mehreren richtigen Symbolen in ein höheres Level wechseln.
* Das Spiel muss mit zunehmendem Level schwieriger werden.

**SOLL-Kriterien:**

* Der Spieler soll zusätzlich akustisches Feedback erhalten.
* Der Spieler soll selbst entscheiden in welcher Schwierigkeitsstufe er spielen möchte.

**KANN-Kriterien**

* Der Spieler kann seinen Spielstand löschen und von vorn beginnen.
* Der Spieler kann aus verschiedenen Spielmodi wählen.
* Das Spiel kann im Modus „Time“ mit Zeitdruck gespielt werden.

## Variable Merkmale des Spiels:

Das Spiel verändert sich während des Spielverlaufs in folgenden Punkten:

* Anzahl der zu merkenden Karten (PlaygroundGrid)
* Anzahl der verdeckten Karten
* Vielfalt der Formen der Symbole
* Vielfalt der Farben der Symbole
* Anzahl der zur Auswahl stehenden Symbole in der unteren Leiste (TilebarGrid)

## Konfigurationen des Spiels:

*In den Einstellungen (*SettingScreen*) kann der Spieler selbst entscheiden, ob er das Spiel mit normalem oder niedrigerem bzw. höherem Schwierigkeitsgrad spielen möchte.*

|  |
| --- |
| SettingScreen: *Spieleinstellungen* |

*Der Einsatz von Polymorphie ermöglichte die einfache Implementierung von verschiedenen Config-Klassen, die auf der normalen Konfiguration (*GameConfig*) aufbauen.*

*Die Schwierigkeitsstufen unterscheiden sich in folgenden Punkten (easy – normal – hard):*

* Größe des Spielfelds (4x3 – 5x3 – 7x4)
* initiale Anzahl der zu merkenden Karten (2 – 3 – 4)
* Anteil der verdeckten Karten (1/3 – 2/3 – (2/3)+1 )
* initiale Vielfalt der Formen und Farben (2 – 3 – 5)
* initiale Anzahl der zur Auswahl stehenden Symbole (2 – 3 – 4)
* Schnelligkeit der Schwierigkeitssteigerung
  + z. B. hard: bei jedem Level-Up erfolgt Schweirigkeitssteigerung
  + s. Player.increase\_level() über verschiedene Modulo-Werte

*Außerdem kann der Spieler seinen gesamten Spielfortschritt zurückzusetzen:*

|  |
| --- |
|  |

# Gameplay und Leveldesign:

Das Spiel ist aus verschiedenen Screens aufgebaut, wobei der MemorizeScreen

und der RememberScreen die wichtigstens Screens für die Interaktionen des Spielers darstellen.

Die folgenden Screenshots zeigen die normale Schwierigkeitsstufe.

|  |
| --- |
| MemorizeScreen: Bildschirm zum Einprägen |

Sobald der Spieler nach eigenem Ermessen die Symbole als „Gemerkt!“ empfindet und den Button betätigt, ändert sich die Ansicht zum RememberScreen. Dieser zeigt dieselbe Symbolreihe an, jedoch ist nun ein Teil der Symbole verdeckt und nicht mehr sichtbar. Eine der verdeckten Karten wird zudem mit einem „?“ markiert, diese Karte muss der Spieler als erstes herausfinden.

|  |
| --- |
| RememberScreen: Bildschirm zur Abfrage |

Zur Auswahl der gesuchten Karte erscheint im unteren Bereich eine bestimmte Anzahl von Karten (TileBarCards). Der Spieler klickt (touched) eine der TileBarCards an und die App prüft, ob sie dem fehlenden Symbol hinter dem „?“ entspricht. Ist dies der Fall, dann ertönt ein positiver Sound und um die (ehemals) gesuchte Karte erscheint ein grüner Rahmen. Gleichzeitig wechselt das „?“ zur nächsten fehlenden Karten und die Auswahl der TileBarCards verändert sich entsprechend.

|  |
| --- |
| RememberScreen: richtiges Symbol wurde ausgewählt |

Bei Auswahl einer falschen Karte ertönt ein negativer Sound, die tatsächlich gesuchte Karte erscheint anstelle des „?“ – nun jedoch mit rotem Rahmen.

|  |  |
| --- | --- |
|  |  |

Nach einer bestimmten Anzahl von richtigen Antworten steigt der Spieler in das nächste Level auf. Der jeweilige Levelfortschritt wird mittels einer ProgressBar im oberen Bereich der Spielscreens angezeigt.

|  |
| --- |
| ScoreHeader zeigt den Levelfortschritt an |

Wenn der Spieler alle „?“ (richtig oder falsch) zugeordnet hat, erhält er eine Anzeige über seinen aktuelle Leistung.

|  |
| --- |
| ResultScreen: Spielergebnis als Indiz für die Merkfähigkeit  oder  etc. |

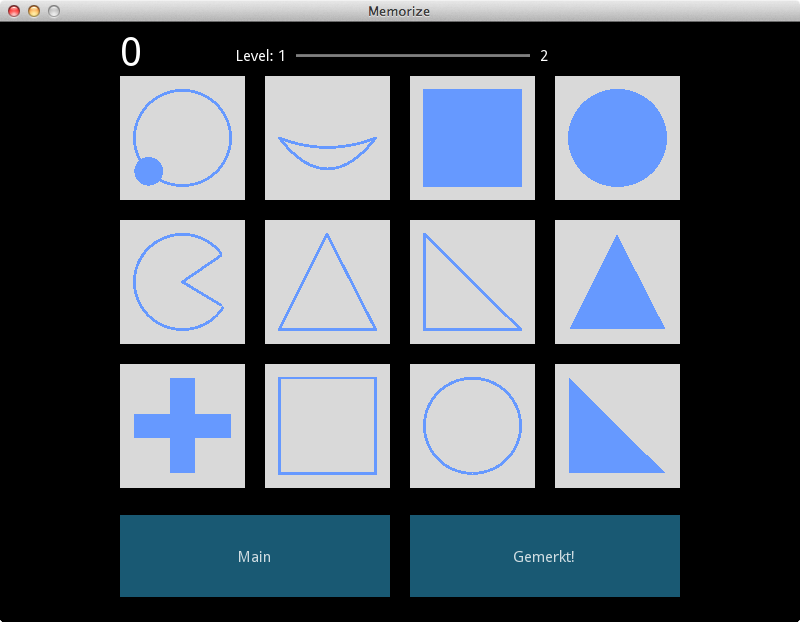
## Speichern des Spielstands:

*Die Speicherung des Spielstands erfolgt in einer JSON-Datei auf dem ausführenden System des Spielers. Diese Datei enthält außerdem Angaben zu den gewählten Einstellungen im SettingScreen (Schwierigkeitsgrad).*

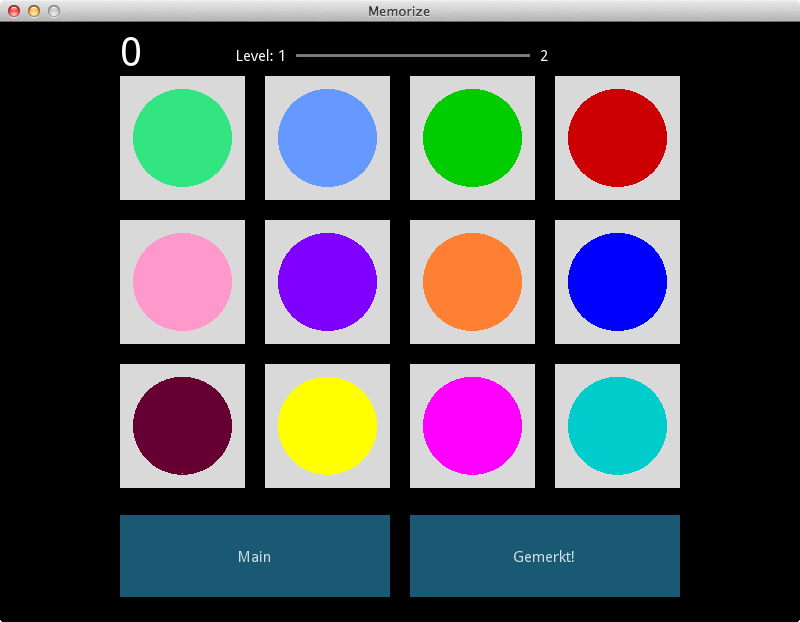
## Implementierte Symbole:

*Mit steigendem Level stehen bei der Erzeugung des Spielfelds mehr und mehr Formen und Farben zur Auswahl.*

* 12 Formen:



* 12 Farben:



# Mögliche Weiterentwicklung der App

In Anbetracht der kurzen Projektdauer und der Neuheit des Frameworks für alle Beteiligten muss das Ergebnis als eine erste Version aufgefasst werden, die noch nicht alle möglichen Funktionen beinhaltet. Dazu zählen die nicht erfüllten Kann-Kriterien aus der Anforderungsanalyse:

* Der Spieler kann aus verschiedenen Spielmodi wählen.
* Das Spiel kann im Modus „Time“ mit Zeitdruck gespielt werden.

Denkbar ist auch, dass das Spiel mit verschiedenen Player-Accounts gestartet werden kann oder mit noch schwierigerer Konfiguration. Die Grundlagen dafür wurden schon jetzt geschaffen, bspw. dank der intelligenten Speicherung im JsonStore und den verschiedenen Config-Klassen.

Ein großer Vorteil von Kivy besteht darin, dass es wegen seiner Python-Basis auch von dessen Portiertbarkeit profitiert, wie bereits in der einführenden Erklärung „Was ist Kivy?“ erwähnt wurde. So wäre es möglich, die vorliegende Kivy-App in eigenständige Anwendungen für andere Plattformen umzuwandeln.