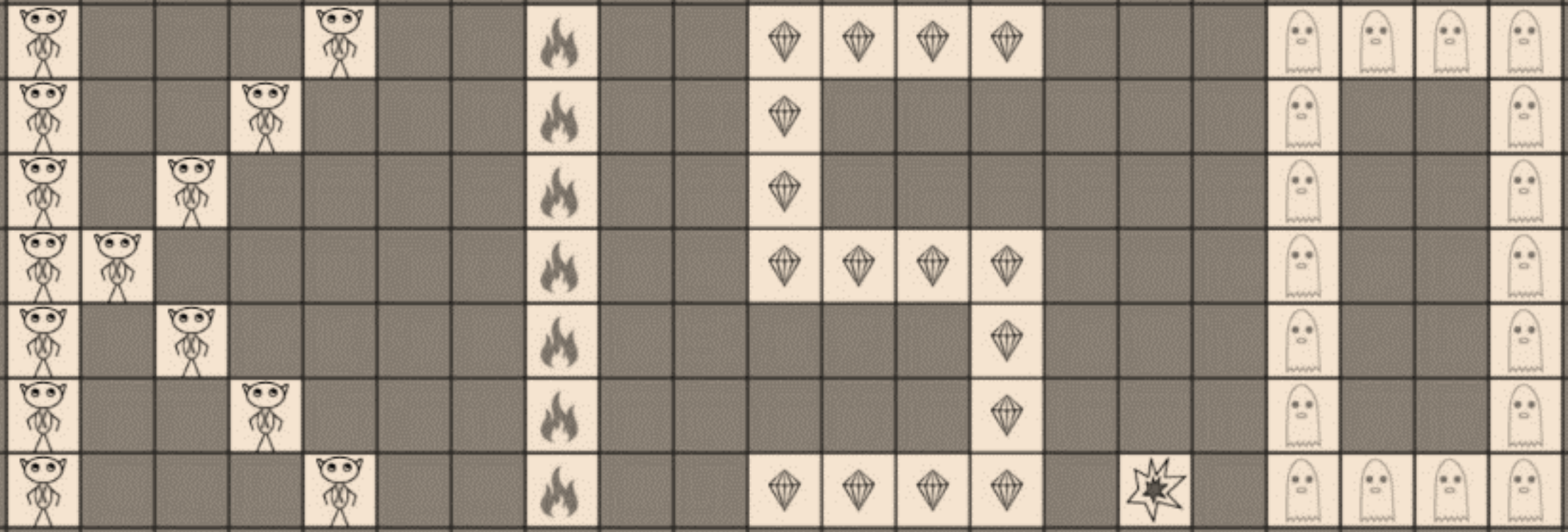
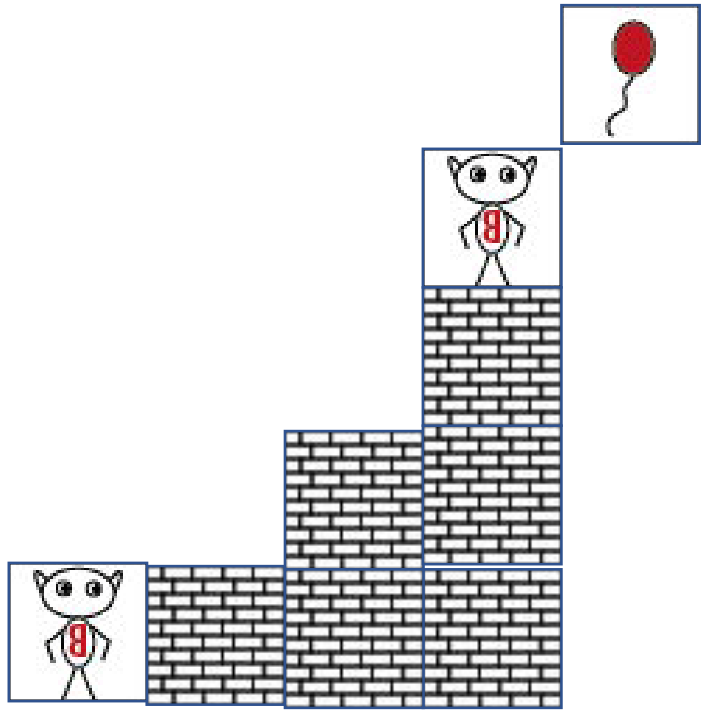
Endpräsentation



Boulder Dash

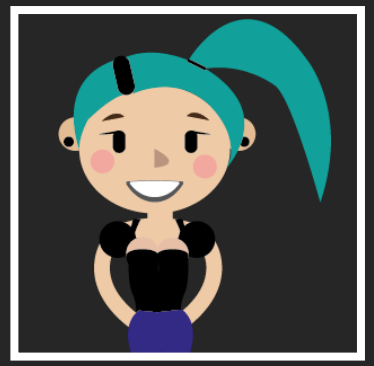
Gliederung

- KI5.0 – Team und Teamwork
- Was kann unser Spiel?
- Die Grundstruktur und Verteilung
- Unsere persönlichen Highlights
- Vorführung



KI5.0 – Team & Teamwork

Players Profiles



Liudmila Kachurina

Parser Queen

„Refactoring!”

Ivana Daskalovska

Level Ruler

„Das ist kein Problem”

Elena Atanasova

Controller Fee

„Ich push das gleich”

Lea Gardner

Rulende Designerin

„Design ist fein”

Elena Rudolph

Logik Queen

„Ohne Bug kein Bugfix“

KI5.0 – Team & Teamwork

Kommunikation:

- Zusätzlich zu Treffen mit Herrn Jost mind. 1 Treffen pro Woche:
- Zum Helfen, Planen, Aufgaben koordinieren & Naschen
- Reger Whatsapp Kontakt

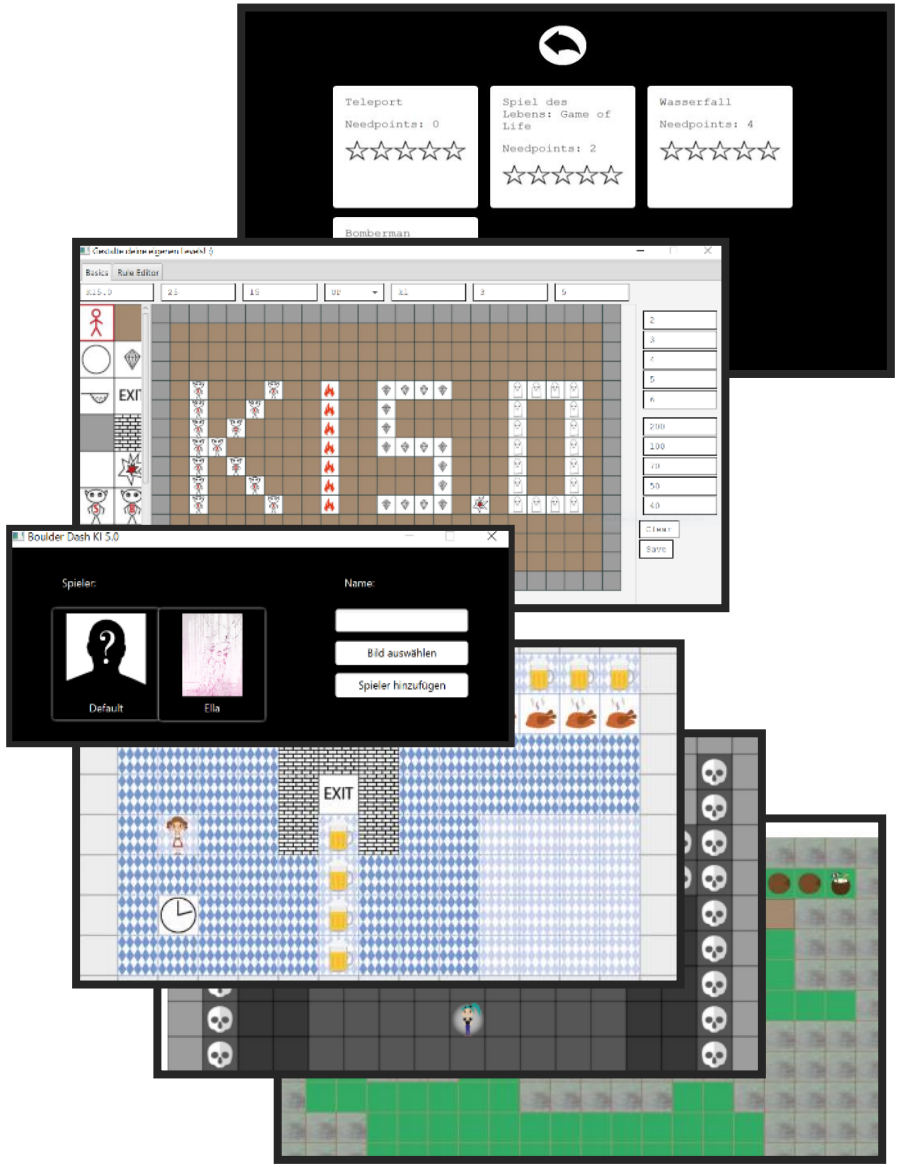
Organisation:

- Aufteilung der Aufgaben
- Setzen von Deadlines

Was kann unser Spiel?

Neben den Mindestanforderungen hat unser Boulder Dash folgende Features:

- Probability, Levelpack
- UI Polish:
  - Mehr als nur ein gutes Design (nämlich 7)
  - mit passendem Sound
  - und tollen Features
- Spielgeschwindigkeit, 5 Sterne
- Spieler Auswahl, Spielstand speichern
- Extrazeit Power-Up
- Mehrspielermodus
- Level Editor (bedingt Rules)

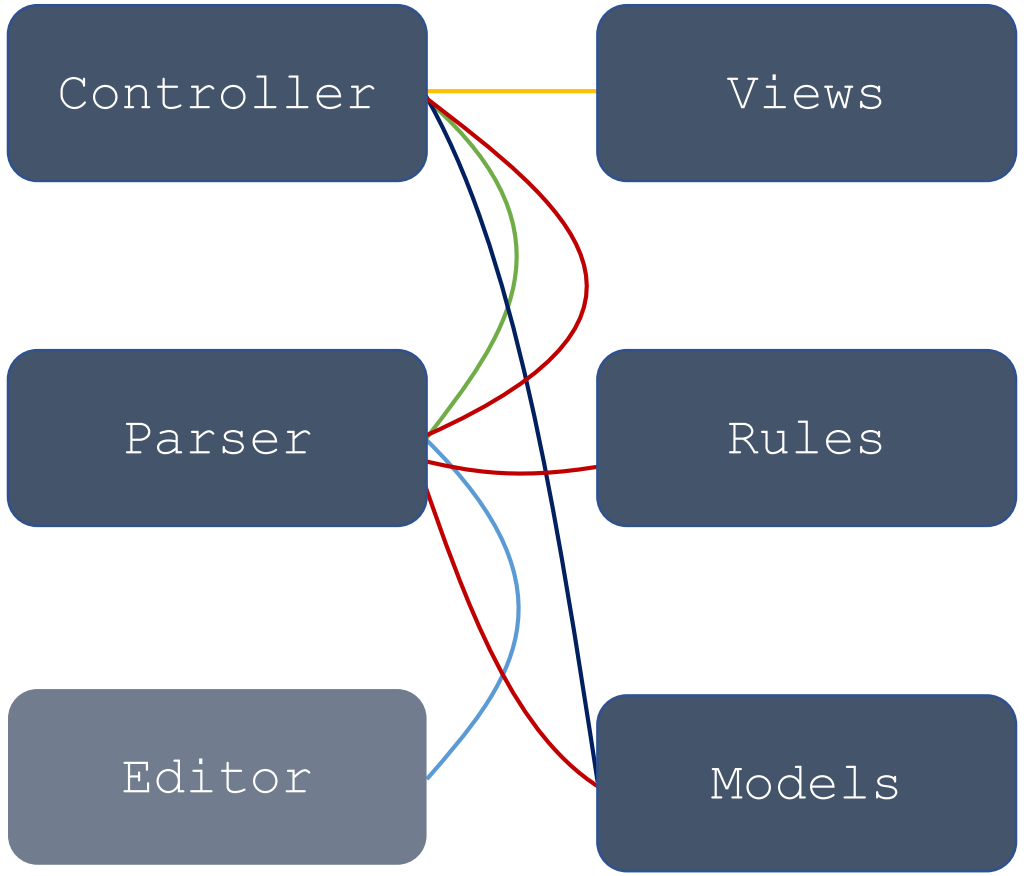


Die Umsetzung: Packages

Steuert die verschiedenen Views und setzt das Model mit **Timeline** in gang.
Läd Medien, Bilder, Json Daten

Parst die Levedaten, sowie unsere Hauptregeldaten mit **GSON.**
Wird vom Editor zum Zurückparsen benutzt

User Interface und eine Klasse die mithilfe des Parsers das Level im Json-Format speichert



Spieler Auswahl, Levelpack Auswahl, Levelpack, Level Ansicht, Ergebnis Fenster

Hauptregeln und Spiellogik:
Bei uns komplett **in Json geschrieben**

Klassen zur Umsetzung der Spielmechanik und zur Verwaltung von Spieldaten.
Grundbausteine: jede Menge **Enums**

Die Verteilung

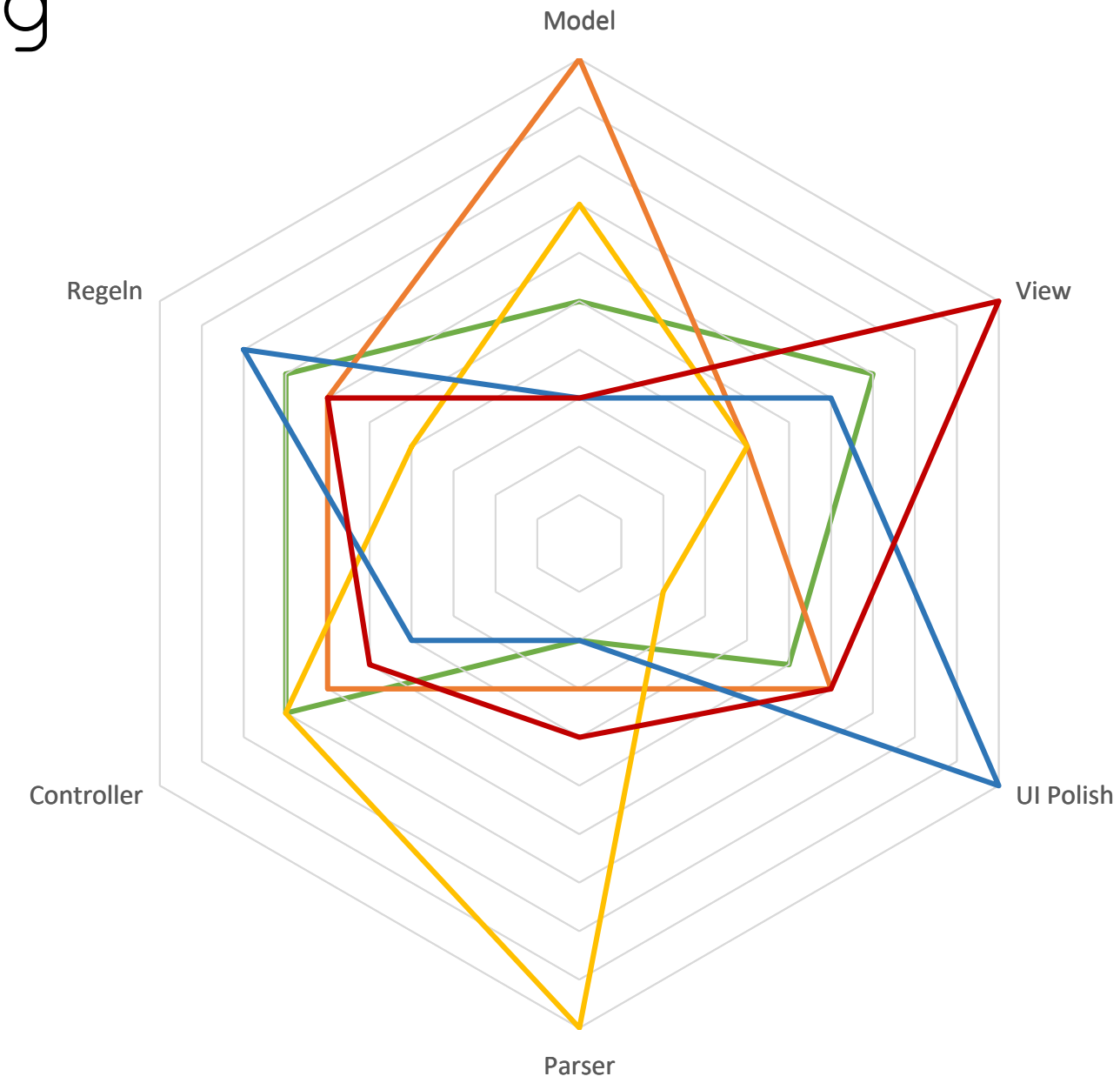
Liudmila Kachurina

Ivana Daskalovska

Elena Atanasova

Lea Gardner

Elena Rudolph



Die Verteilung

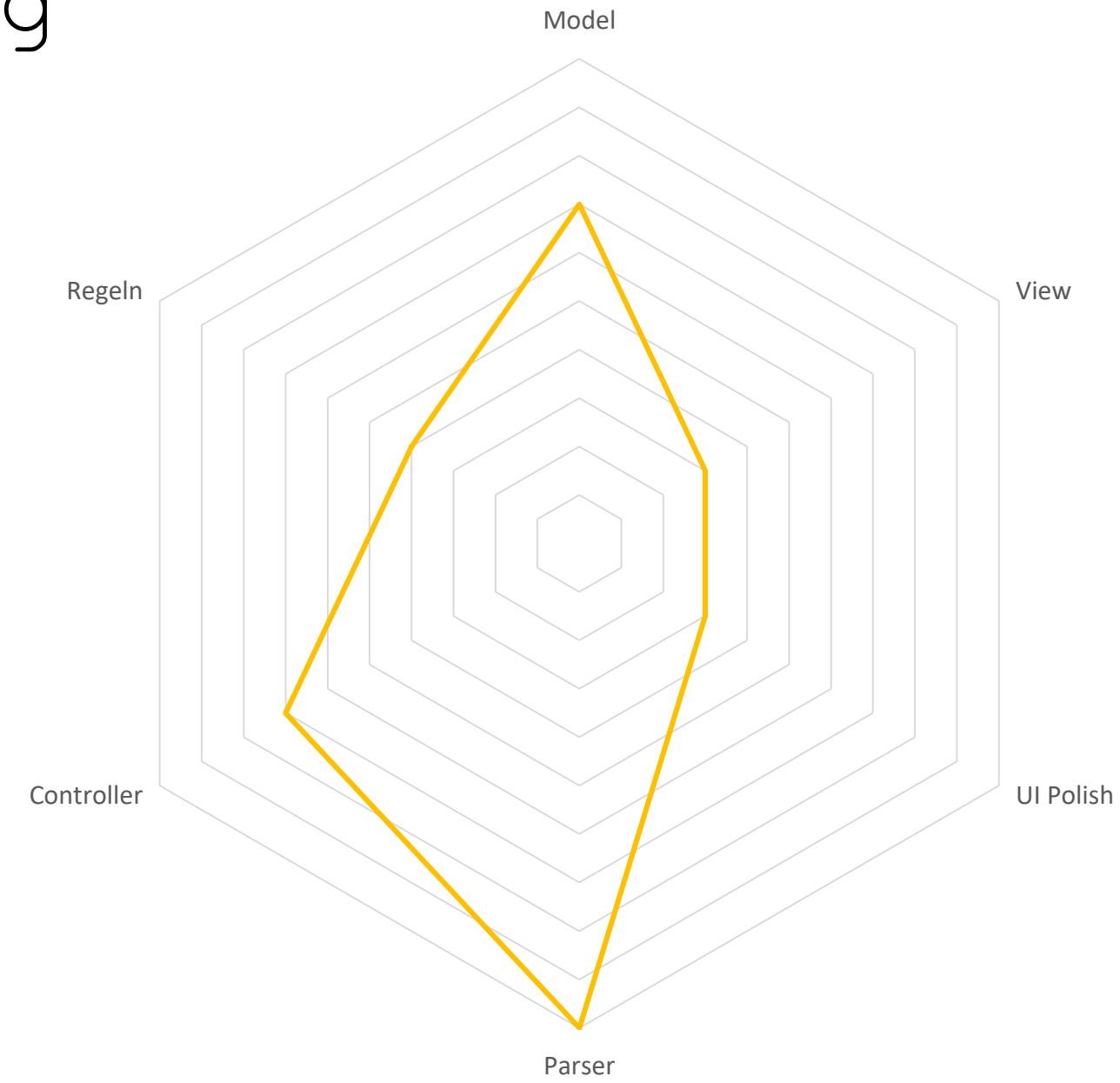
Liudmila Kachurina

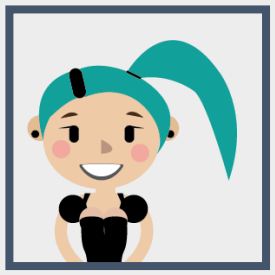
Ivana Daskalovska

Elena Atanasova

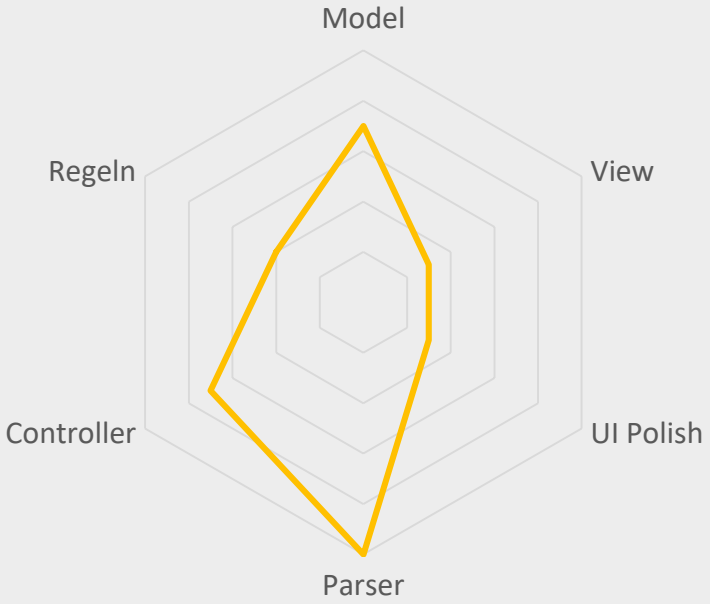
Lea Gardner

Elena Rudolph





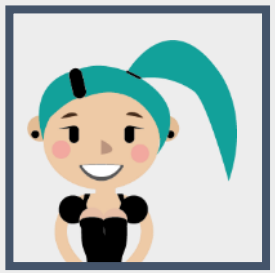
Liudmila Kachurina



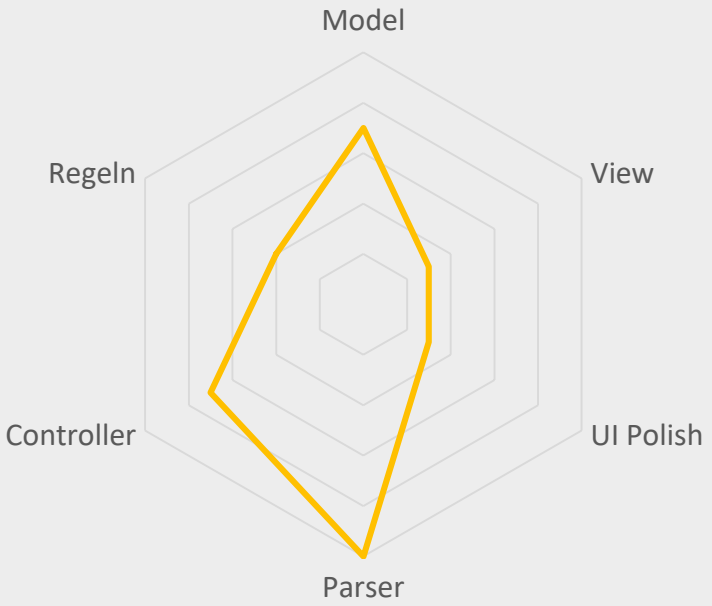
Highlights

Allgemein: kompletter Parser, Controller, Model

- Refractoring des ganzen Programms und Debugging
- Besonderheit: Aufbau des Parsers mit **Gson**-Bibliothek
- Problem: Backup von Levels
  Lösung: Komposition LevelData & Level gelöst.
  - Beim Öffnen eines Level-Paketes werden Level-Metadaten geladen
  - Beim Neustarten eines Levels -> Level vollständig gelöscht , in Metadata wird ein neues Level erstellt -> Speicherung vom Spielprogress
  - Beim Übergang zu anderen Level-Paket werden alte Metadaten gelöscht und neue geladen→ so nie out of memory

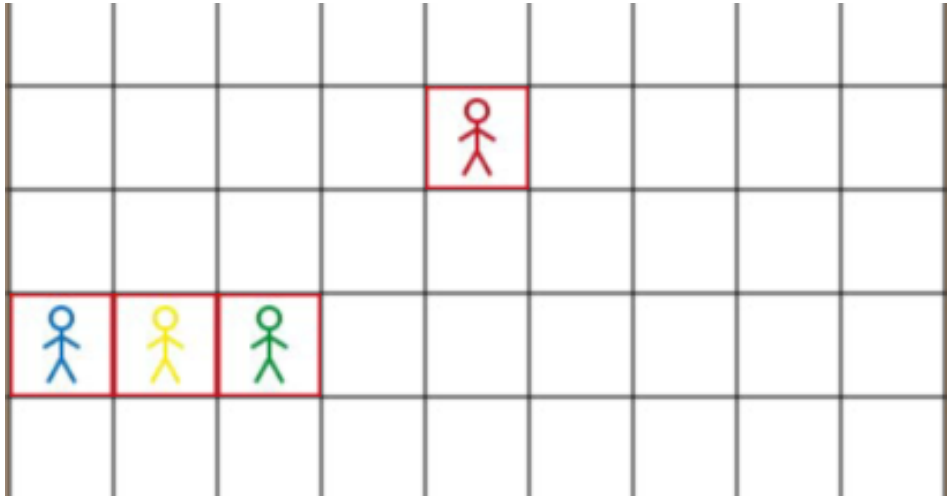


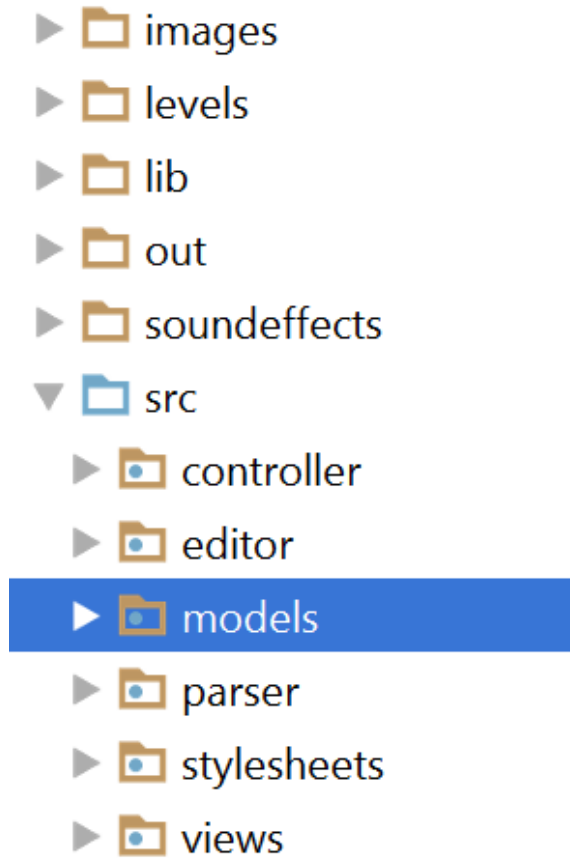
Liudmila Kachurina



Highlights

- **41 Klassen+ jede Menge Files:** in sinnvolle, effiziente Struktur gebracht
- Mehrspielermodus





Die Verteilung

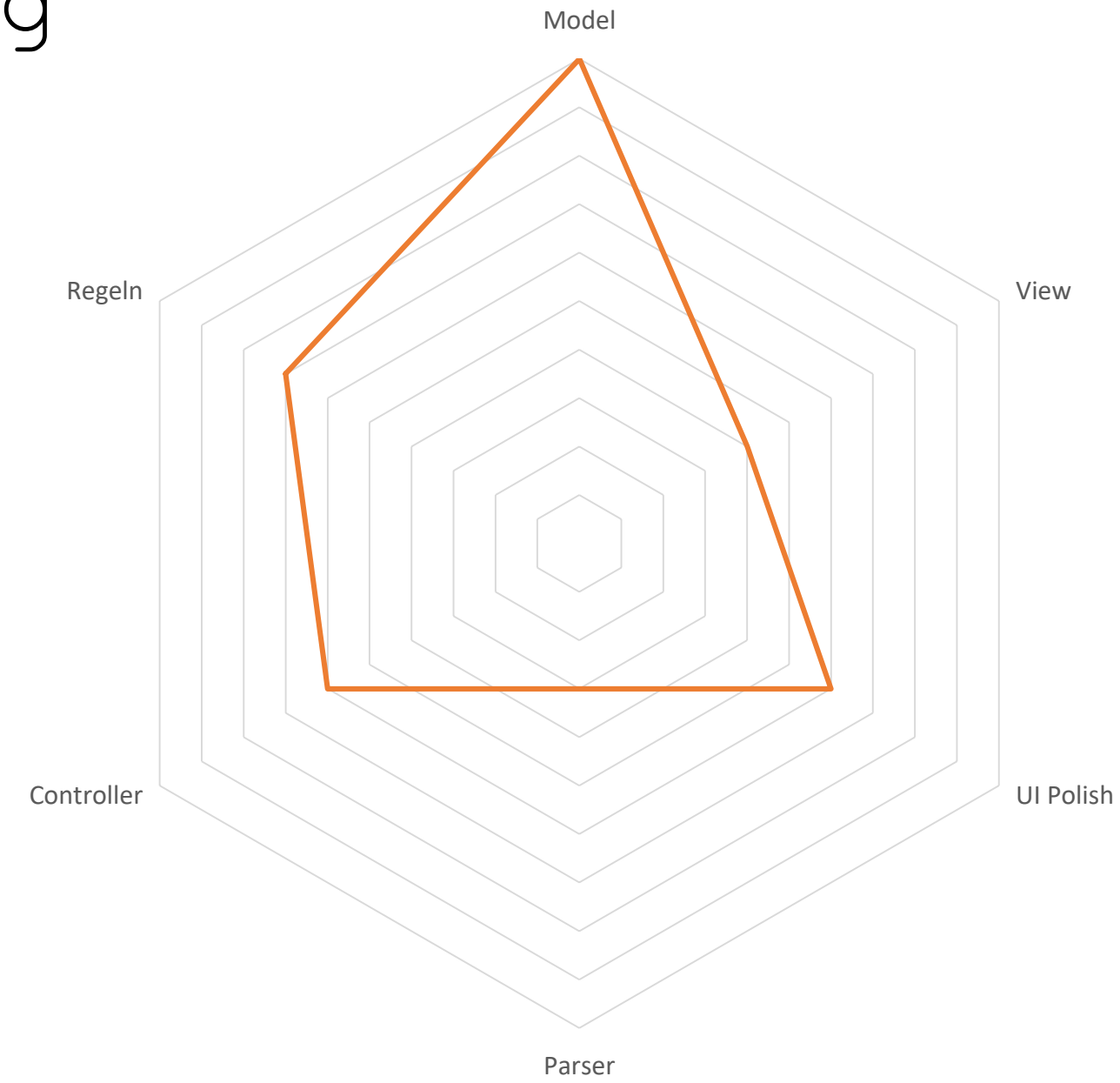
Liudmila Kachurina

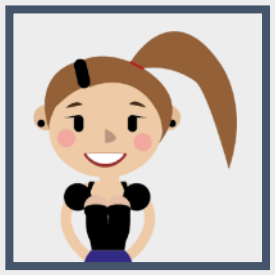
Ivana Daskalovska

Elena Atanasova

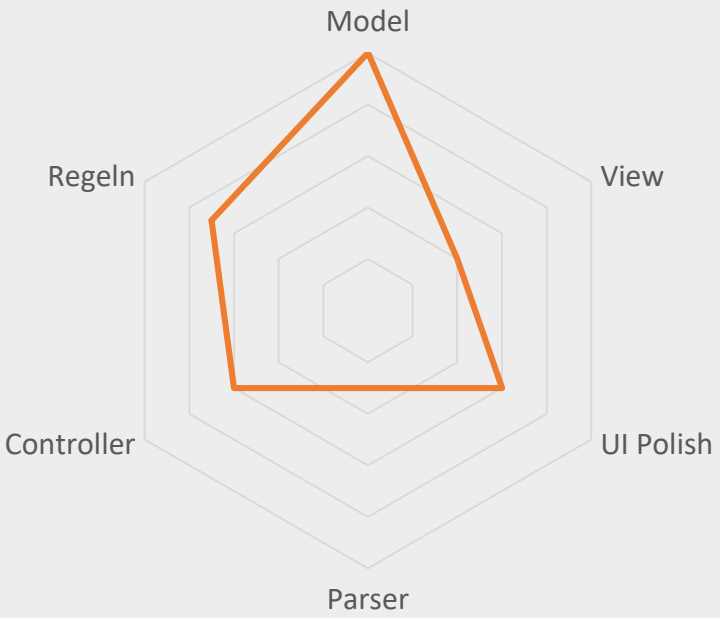
Lea Gardner

Elena Rudolph



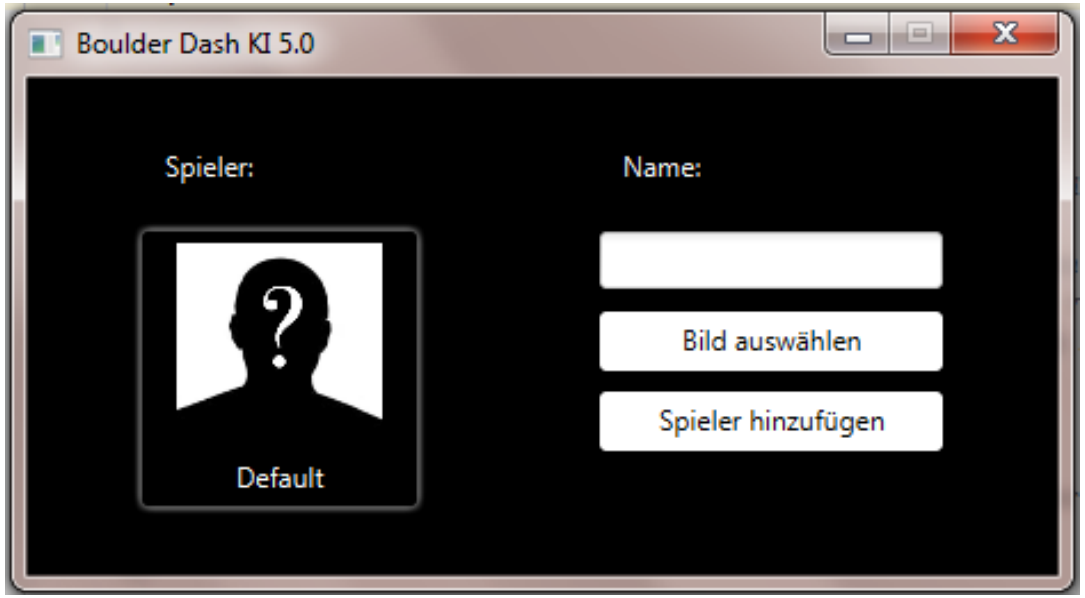


Ivana Daskalovska

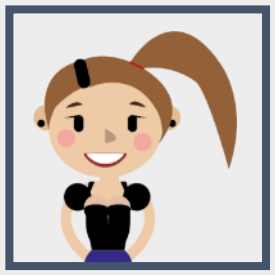


Highlights

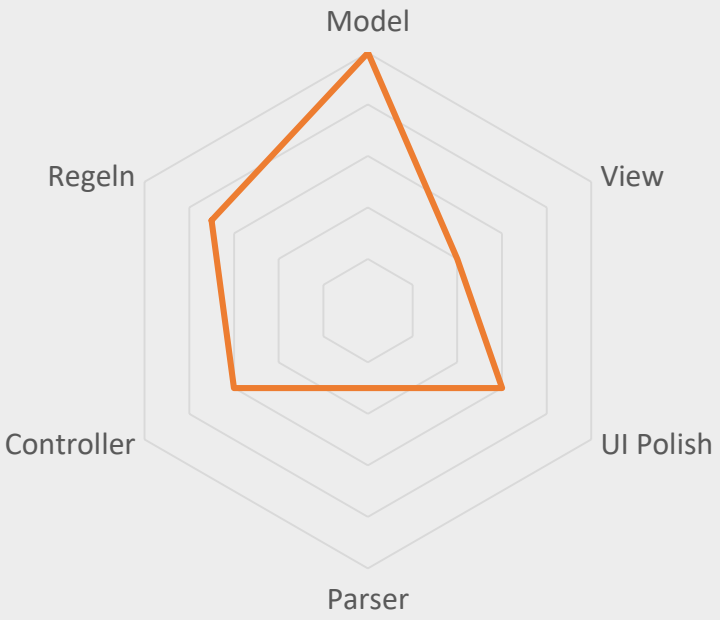
Allgemein: Modelklassen, 5 Sterne, Spielstand speichern, Spielgeschwindigkeit, Sperren & Freigeben von Levels



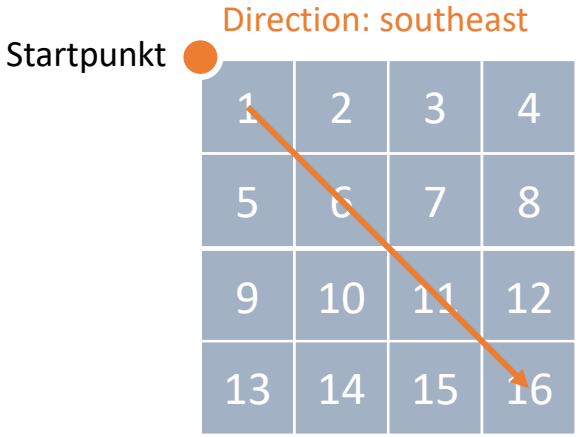
- Spieler Auswahl (Spielstand Speicherung)
- Serialisierung und Deserialisierung von der Klasse LevelData



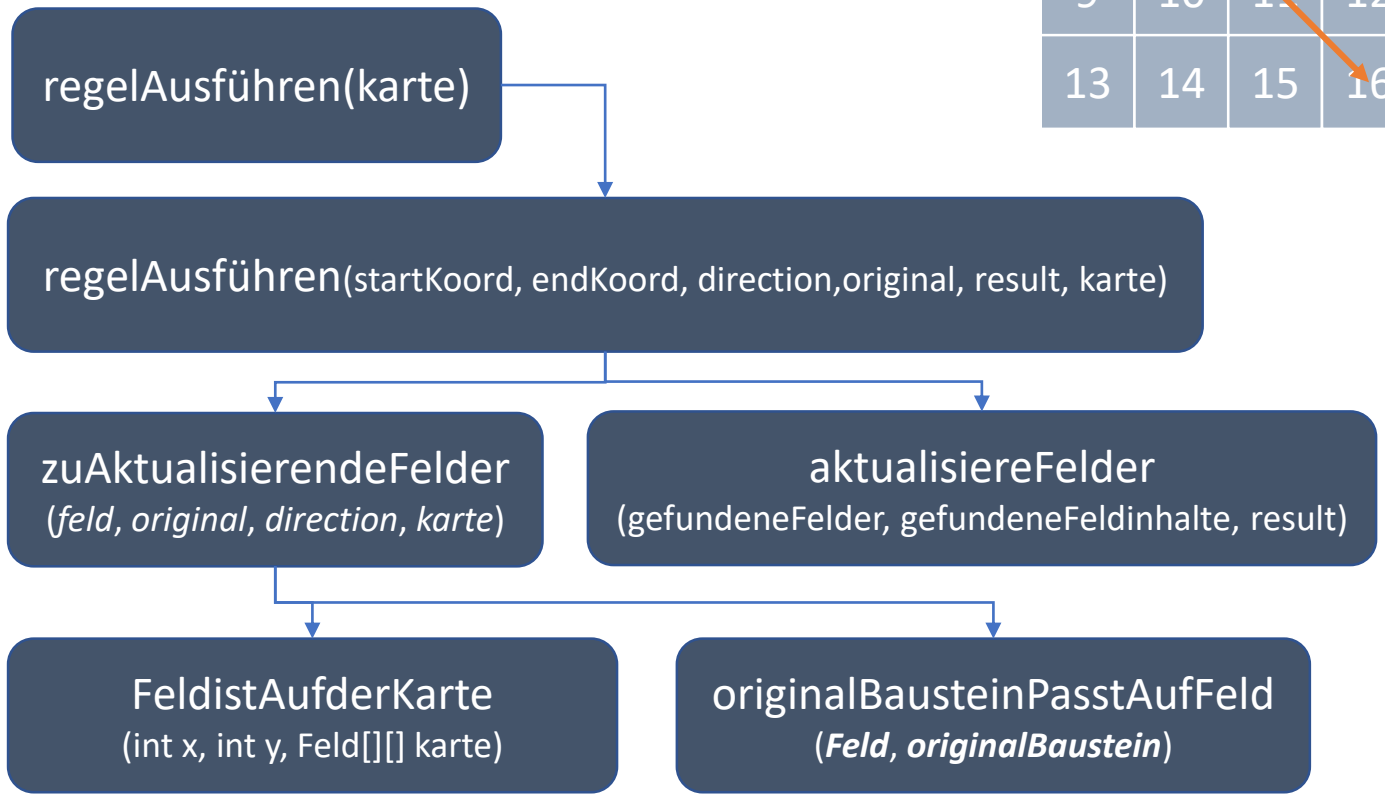
Ivana Daskalovska



Highlights



Regel Ausführung



Die Verteilung

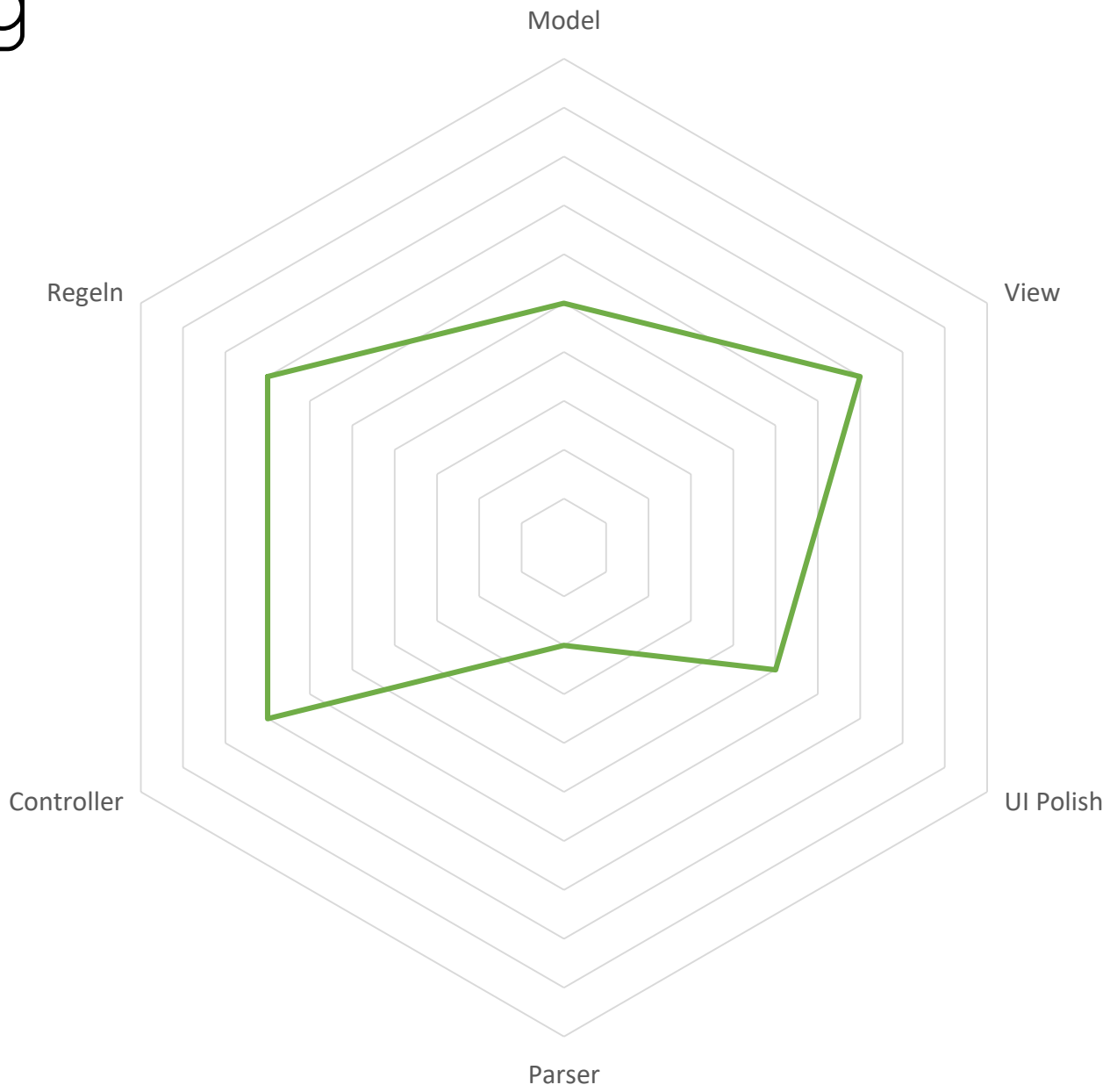
Liudmila Kachurina

Ivana Daskalovska

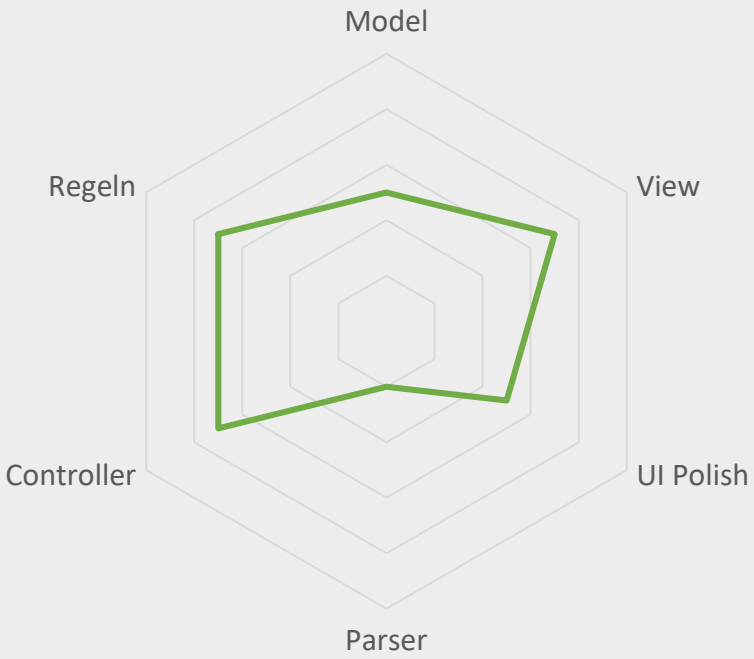
Elena Atanasova

Lea Gardner

Elena Rudolph



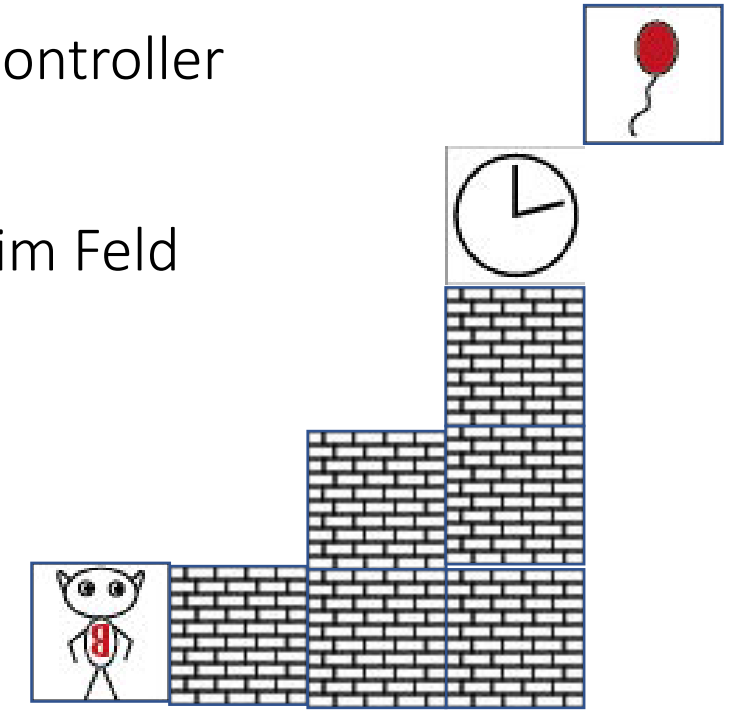
Elena Atanasova



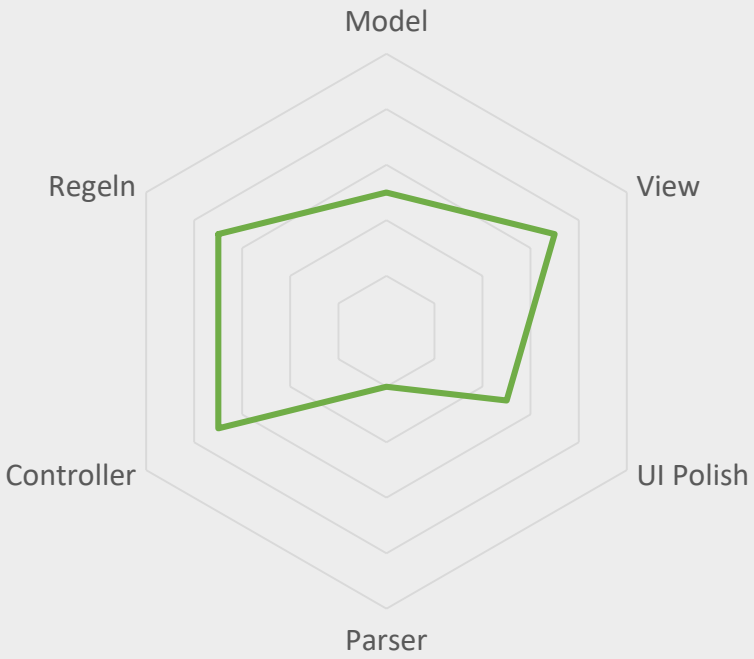
Highlights

Allgemein: Timeline & Tickablauf, PackAuswahl, Ergebnisfenster, UI-Polish, PlayerModusAuswahl

- UI Polish Elemente: Sterne, Extratime, Pause, CSS
- Timeline Steuerung im ModusController
- macheTick() im Level
- Flags zurücksetzen, im Level -> im Feld

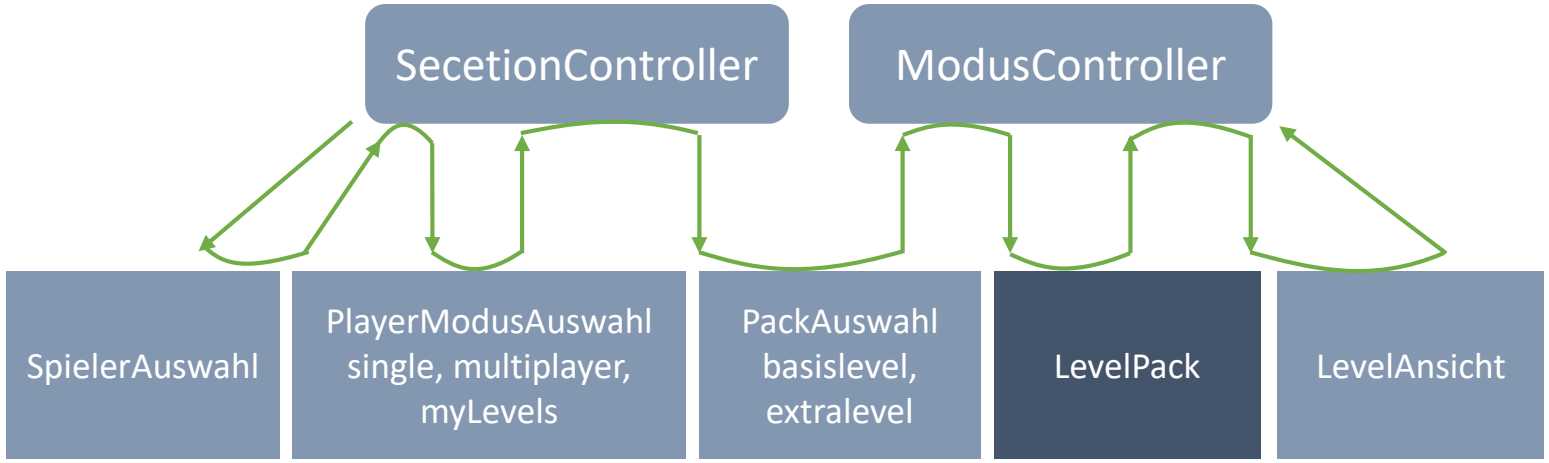


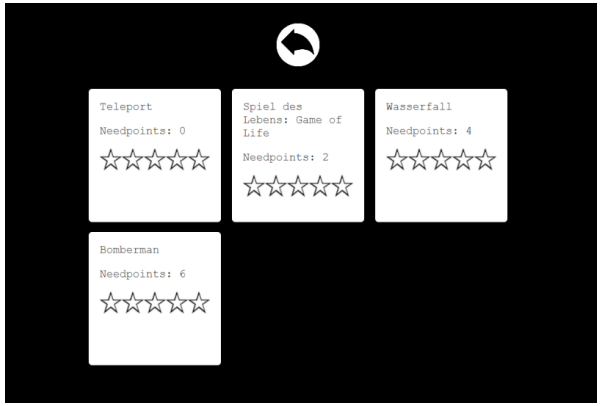
Elena Atanasova

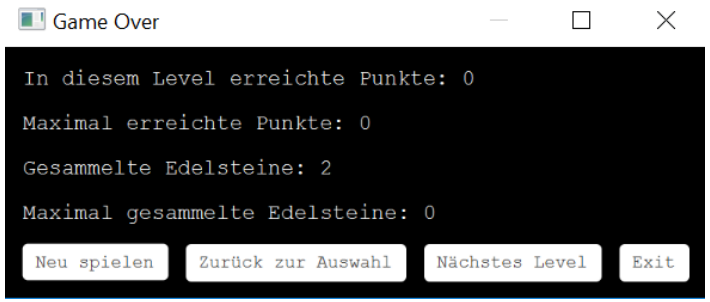


Highlights

- Level Auswahl und Ergebnisfenster







Die Verteilung

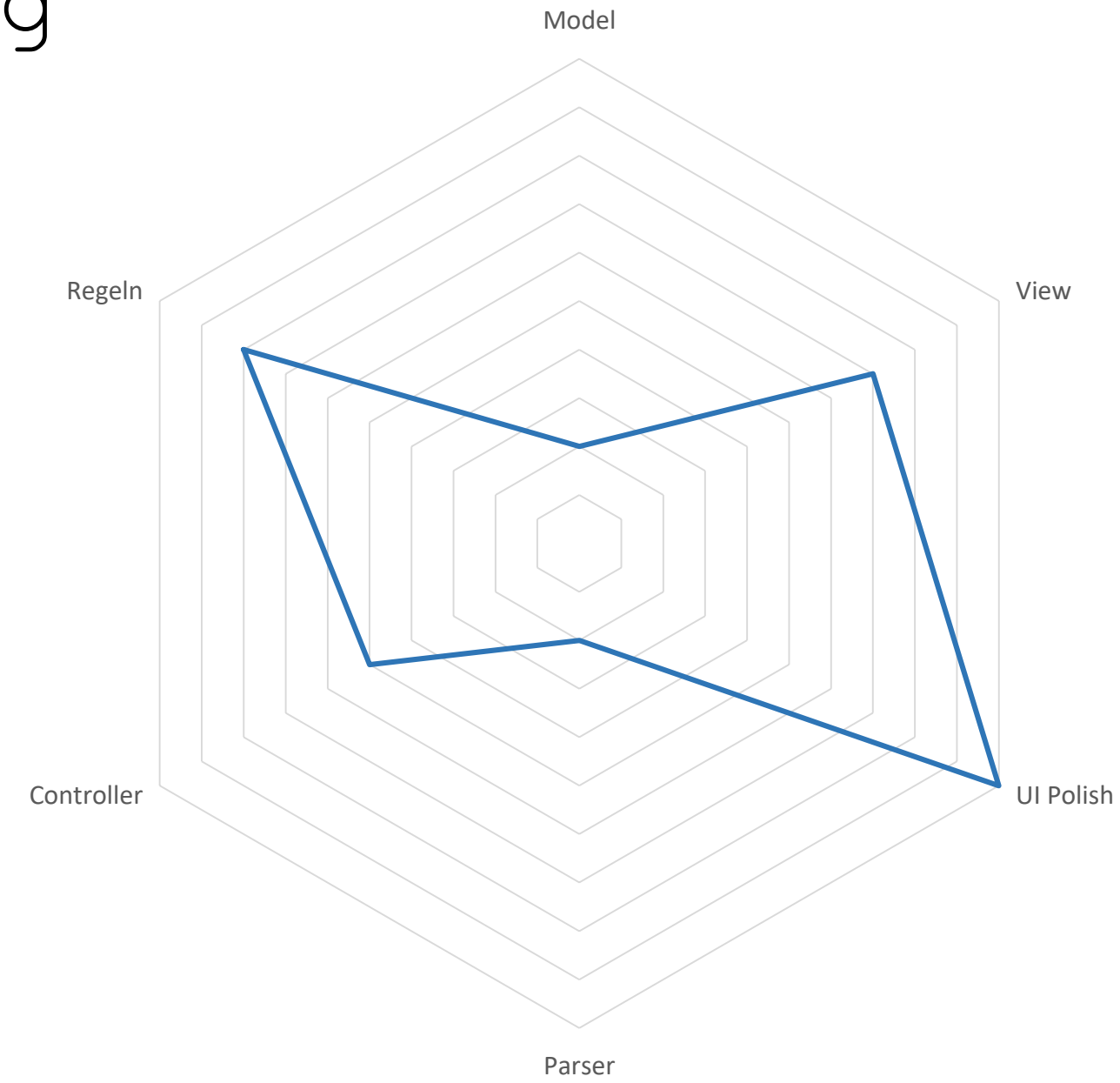
Liudmila Kachurina

Ivana Daskalovska

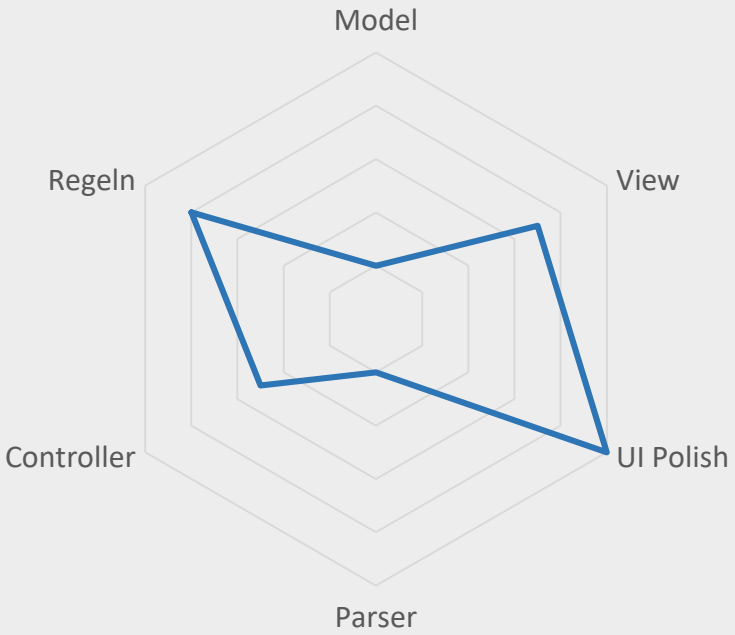
Elena Atanasova

Lea Gardner

Elena Rudolph



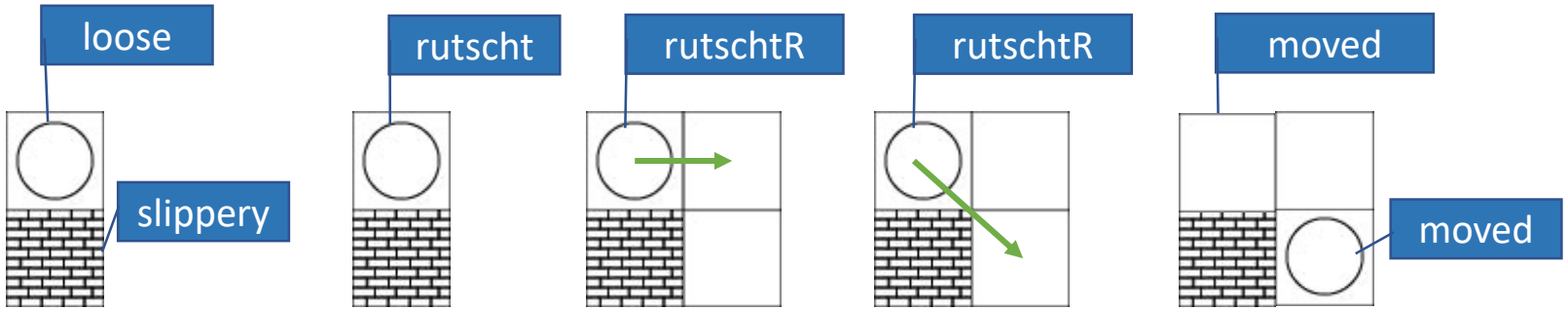
Lea Gardner



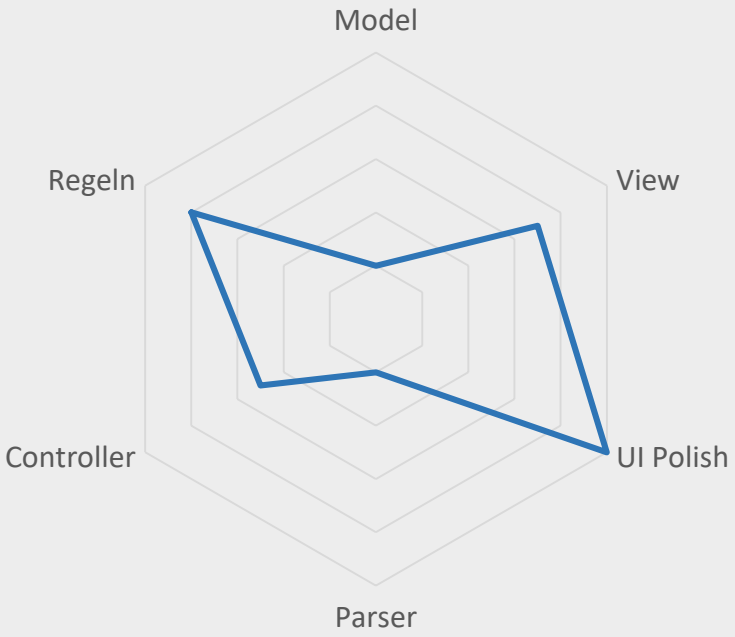
Highlights

Allgemein: Hauptregeln, View Elemente, Model Elemente gesamte Grafik, Soundimplementierung

- Soundloader wird zusammen mit Levelansicht geladen
  - Dabei wird Design aus Json Levelmetadaten gelesen
  - Sound passend zum design
- Vorteile von Hauptregeln in Json:
  weniger Javacode, übertragbar, bleibt unberührt
- Json Beispiel: Objekt Rutscht

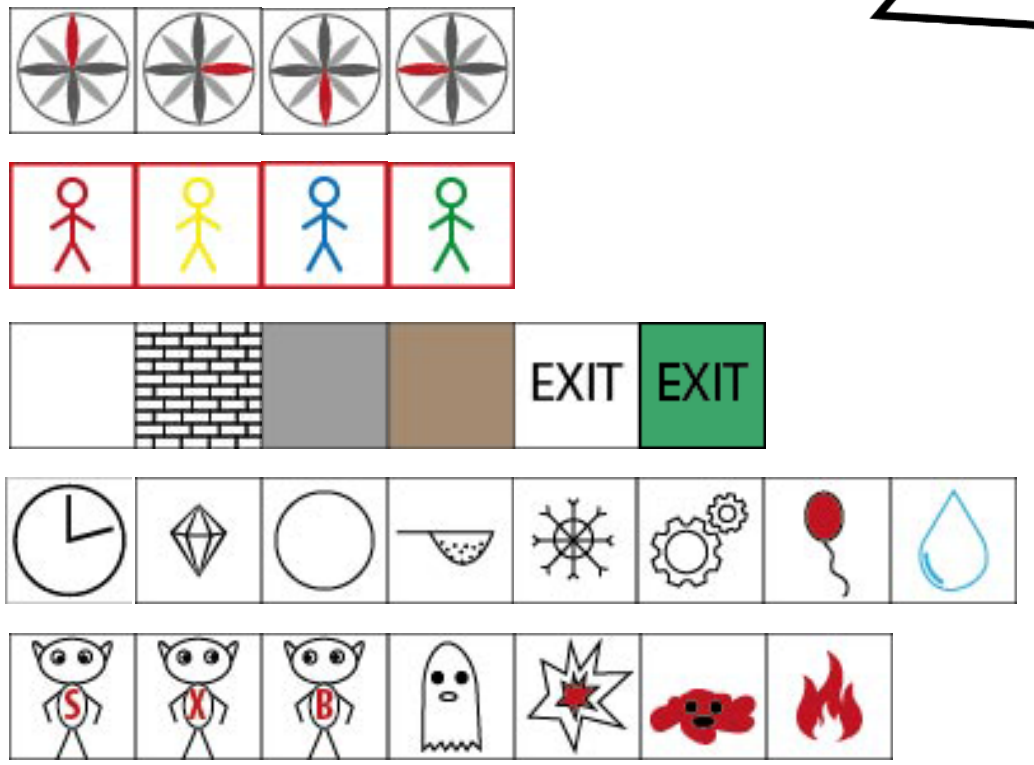


Lea Gardner

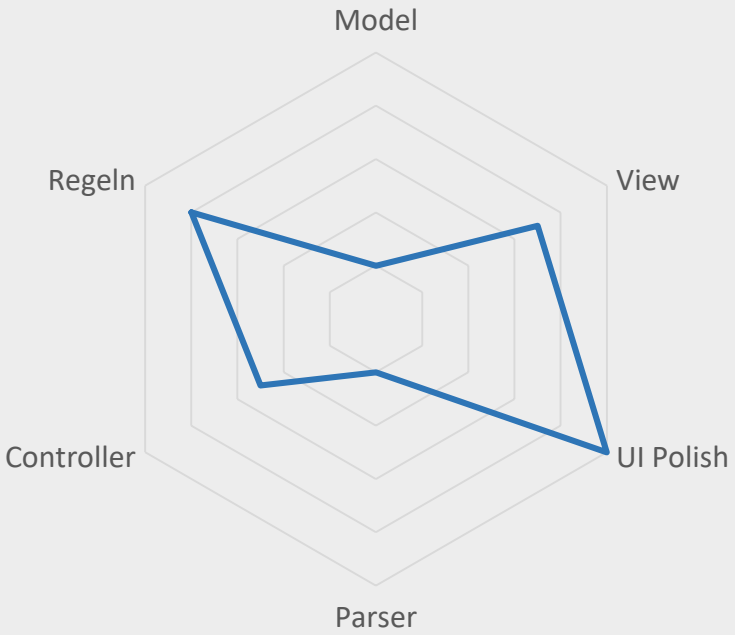


Highlights

- Default- Design:
  Klares Design ohne Storytelling



Lea Gardner



Highlights

- Verspieltes Design mit Storytelling:

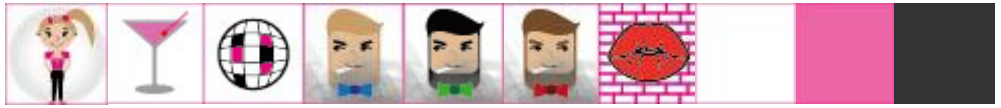
Bayrische Boazn

Verbotenes Konzert

Festival

Jamaica Party

Metal Party



Schiki Miki

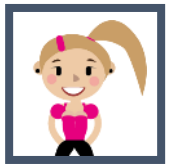
Die Verteilung

Liudmila Kachurina

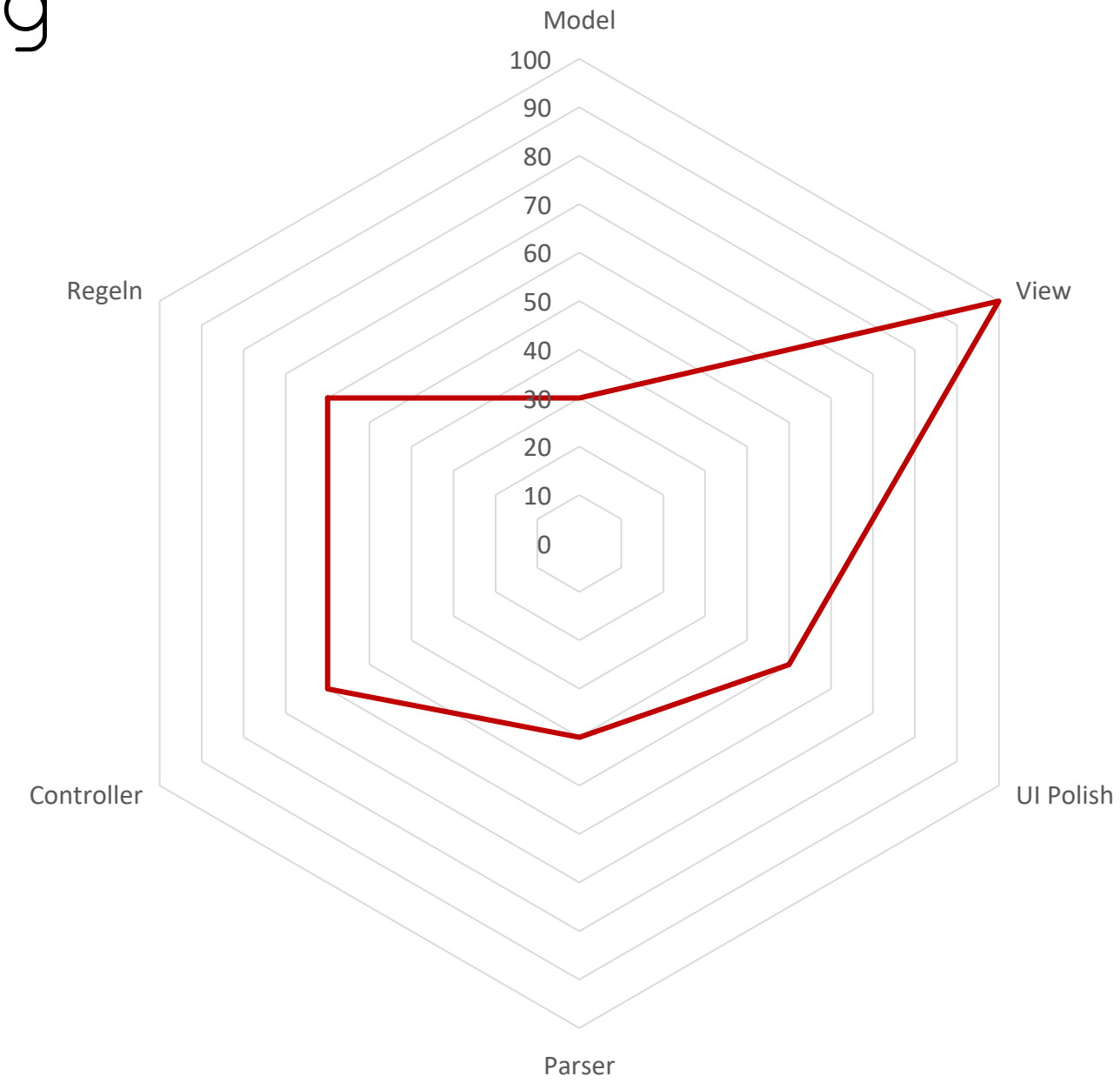
Ivana Daskalovska

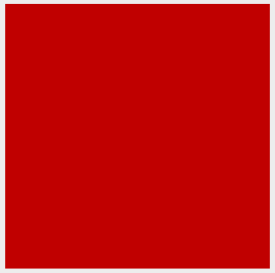
Elena Atanasova

Lea Gardner

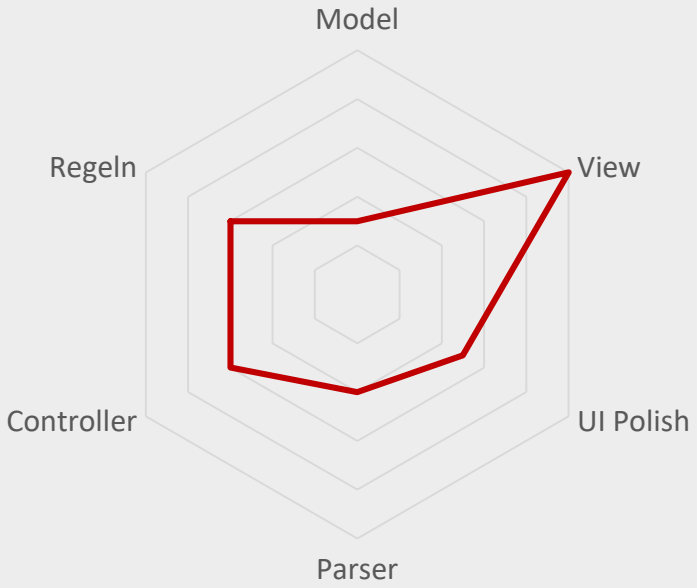


Elena Rudolph





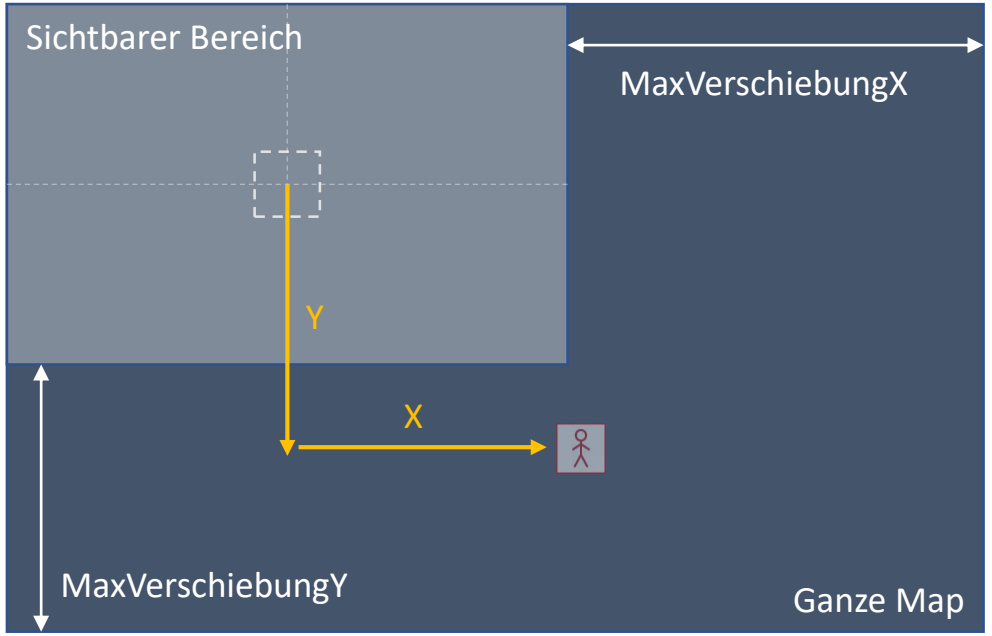
Elena Rudolph

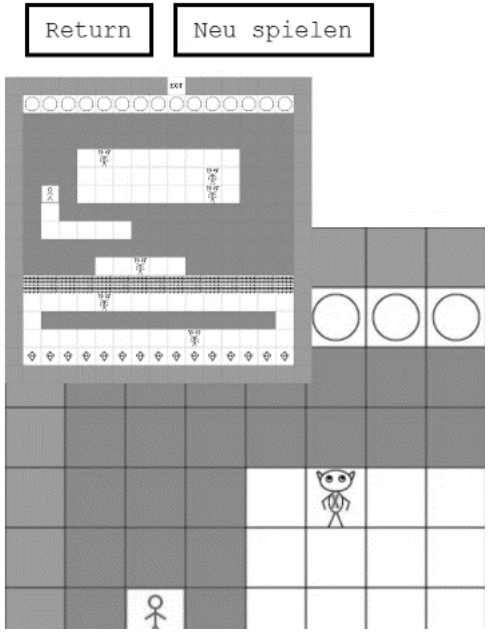


Highlights

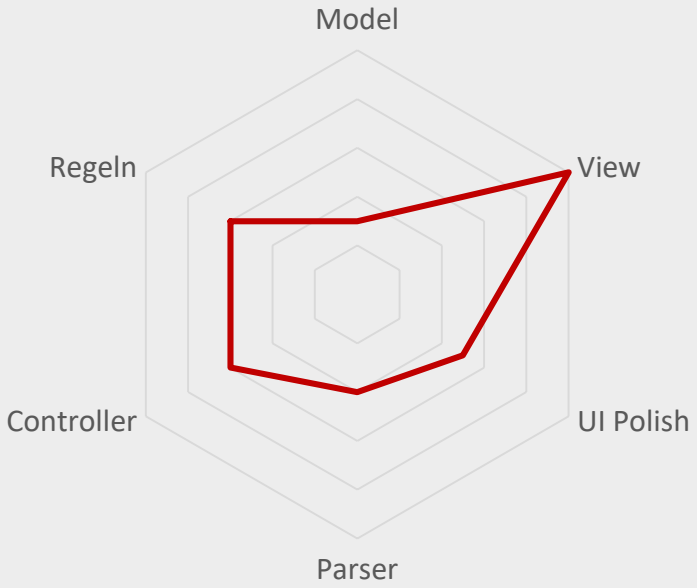
Allgemein: Levelansicht, erste Levelauswahl, erster Controller, EventHandler, Imageimplementierung, Level Editor, Regelfixing

- Scrolling und Minimap
  jeder Tick ein neues Translate(x, y)



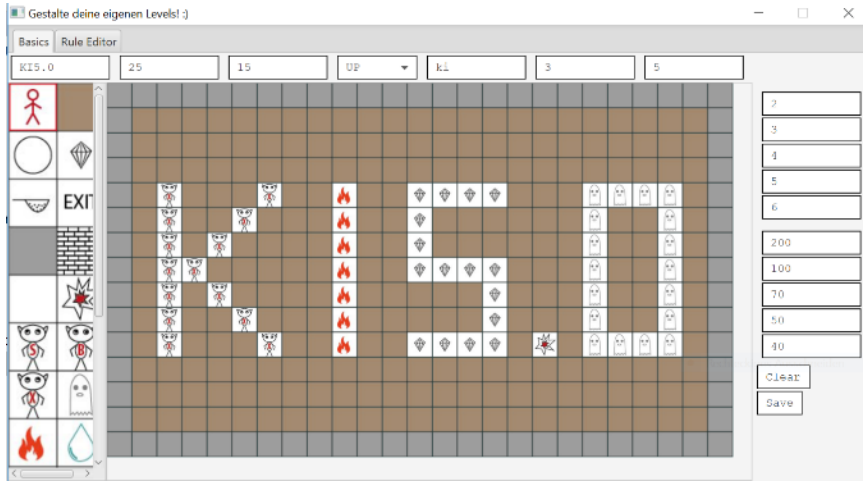


Elena Rudolph



Highlights

- LevelEditor: Erstellen und abspeichern von Levels die direkt ins System eingespeist werden



Problem des Rule- Erstellens: Hauptsächlich Zeit
Für abspeichern in Json sollte Parser Klasse genutzt werden -> nicht 100% geeignet
Workaround: Level komplett aus Json Objekten zusammensetzen

- ImageLoader
  - Laden von Bildern beim LevelAnsicht-Start
  - Initialisierung mit Levelspezifischem Design Path
  - getImages(Feldinhalt) → Passendes Image kommt heraus

Vielen Dank für Ihre Aufmerksamkeit!

Demo – Time!