Факултет за информатички науки и компјутерско инженерство (ФИНКИ)

Тимски Проект

Candy Crush Clone со адаптивна тежина преку Вештачка Интелигенција

Ментор:
д-р Катарина Тројачанец Динева

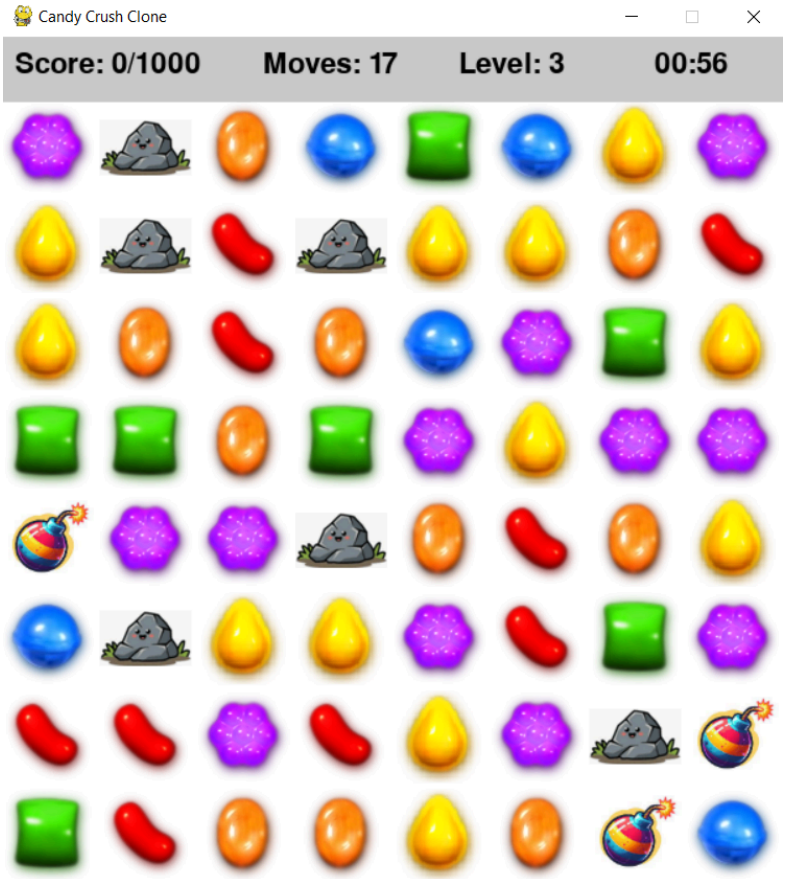
Членови на тим:
Викторија Страторска 213011
Ива Макенаџиева 213013
Миле Ѓоргиев 211093
Кристина Стефанова 213103
Кире Поповски 211222

Опис и цел на проектот:

Овој проект е реплика на популарната игра Candy Crush, изработена со програмскиот јазик Python и библиотеката Pygame. Целта е да се комбинираат повеќе исти бонбони во ред или колона за да се добијат поени. Секое ниво има одреден број на поени кои мораат да се соберат со ограничен број на потези и во одредено време. Играта вклучува повеќе нивоа со различни тежини, блокери и бомби за кои тежината прогресивно расте - секое следно ниво е потешко од претходното. Дополнително, има модул за вештачка интелигенција кој ја следи успешноста на играчот и ја прилагодува тежината во следните нивоа.

Користени технологии:

- **Python** – главен програмски јазик
- **Pygame** – за графика, анимации и управување со настани
- **JSON** – за зачувување податоци (име и најдобар резултат на играчот)
- **Сопствен AIModule** – модул за адаптивна тежина



Главни функционалности:

- Динамична тежина за секое ниво (во зависност од перформансите во претходното ниво)
- Различни типови на полиња: бонбони, блокери и бомби
- Анимации при размена и паѓање на полиња
- Автоматско препознавање и одстранување на направени комбинации
- Динамично сетирање на тајмер и ограничен број потези
- Приказ на резултат, ниво и останати потези
- Почетен екран, екран за премин на ниво и екран за Game Over
- Зачувување и читање на податоци од фајл

- Поврзување на бонбони

  Хоризонтално или вертикално поврзување на 3+ исти бонбони
  Повеќе бонбони = повеќе поени

- Анимации

  Паѓање на нови бонбони
  Анимација на замена (swap)

- Специјални елементи

  Блокатори – се појавуваат на повисоки нивоа и го отежнуваат играњето
  Бомби – се активираат ако се во близина на совпаѓања и одземаат поени

- Секое ниво има:

  Цел (target score)
  Ограничен број потези
  Временско ограничување

- Тежината се зголемува со:

  Повеќе блокатори
  Повеќе бомби
  Помалку потези

- AI модул за динамичка тежина

  Следи перформанси на играчот (брзина, преостанати потези)
  Прилагодува ја тежината на следното ниво (промена на бројот на потези/време)

- Зачувување на податоци

  JSON датотека (`save_data.json`) ги зачувува:

  Името на играчот
  Највисокиот резултат

Структура на кодот:

- **Главни класи:**
  - **`GameState`** – ја управува логиката на играта (нивоа, поени, мрежата)
  - **`AIModule`** – анализира перформанси и прилагодува тежина
- **Главни функции:**
  - **`check_matches()`** – проверува дали има совпаѓања на бонбони
  - **`remove_matches()`** – ги брише совпаднатите бонбони и доделува поени
  - **`fill_empty_spaces()`** – пополнува празни места со нови бонбони
  - **`handle_swap()`** – управува со заменувањето на бонбони

Дизајн:

- **Бонбони** (6 бои)
- **Блокатори** (сиви камења)
- **Бомби** (црвени со фитил)
- **Текст и UI** (поени, нивоа, време)

Вештачка интелигенција (AI)

Модулот AIModule собира информации за играчот и врз основа на нив ја одредува тежината на следното ниво:

- Поголем резултат и помалку време → потешко следно ниво

- Помал резултат → полесно следно ниво

Во овој проект, вештачката интелигенција користи **Linear Regression** модели од библиотеката scikit-learn за адаптивно менување на тежината на секое следно ниво.

Користени AI модели:

Се користат два модели на линеарна регресија:

- **move_model**: одредува **лимитот на потези** (move_limit)
  предвидува колку потези ќе бидат потребни за следното ниво.
- **time_model**: одредува **временскиот лимит** (time_limit)
  предвидува колку време ќе биде потребно за играчот да го заврши нивото.

Со овие модули се прилагодува тежината на следното ниво. Тоа се врши преку методот calculate_difficulty(), со што играта станува пополесна или потешка, зависно од претходните резултати на играчот.

Податоци за тренирање:

AI модулот собира податоци за успешноста на играчот, како што се:

- број на ниво (level)
- искористени потези (moves_used)
- постигнат резултат (score)
- потрошено време (time_used)

Овие податоци се зачувуваат со секое успешно завршено ниво. AI ги користи за да ги тренира моделите преку функцијата train_models().

Имплементација:

- **Pygame иницијализација**

  Го поставува екранот (600x650 пиксели).
  Големината на таблата е 8x8 (секој бонбон е 75x75 пиксели).

- **Исцртување на бонбони / бомби / блокери**

  Секој елемент во играта како што се бонбоните, блокерите и бомбите е **слика** (ако постои) или **цртан правоаголник** (ако не е пронајдена сликата).

- **Креирање и чување на податоци за играчот**

  Доколку немаме никакви податоци за играчот, односно не постои save_data.json фајл, името на играчот кој тој ќе го внесе преку инпут поле кое се појавува на почетниот екран се зачувува во нов JSON фајл, заедно со неговата highscore вредност. За играчи кои повторно ја играат играта нивните податоци се вчитуваат од веќе постоечка JSON датотека, името на играчот се прикажува на почетниот екран и се ажурира вредноста за highscore доколку има промена.

- **Game State класа**
  Со помош на разни функции во game state класата се справуваме со целата логика на играта.

  Освен вчитувањето на бонбоните, на одредни нивоа играчот се соочува и , со блокери и бомби кои се поставуваат преку *place_blockers_for_level()* и *place_bombs()* фукнциите.

  Со цел да нема помесување на елементите пред играчот да го направи својот потег, *ensure_no_matches_at_start()* функцијата проверува дека не постојат никакви совпаѓање на бонбоните, а доколку се појават некои совпаѓања се врши ново поставување на бонбоните. Оваа функција ја повикува *check_matches()* функцијата која вертикално и хоризонтално проверува дали има совпаѓања на три или повеќе елементи од иста боја и сите совпаѓања ги враќа во една листа.

  Check_bomb_adjacent() ги разгледува сите воочени совпаѓања и проверува дали до некои од нив има бомба, доколку најде бомба поставена до тие совпаѓања тогаш на играчот му се одземаат дел од стекнатите поени. Оваа фукнција се извршува на почетокот на *remove_matches()* функцијата и откако ќе се заврши со нејзината проверка сите валидни совпаѓања на елементите се тргаат од екранот и се ажурира тековниот резултат (score) на корисникот. Со цел отстранувањето на елементите од екранот да изгледа убаво се извршува едноставна анимација преку *handle_falling_tiles()* функцијата.

Совпаѓањата се случуваат само доколку играчот успешно ги смени местата на бонбоните со цел да дојде до поврзување на три или повеќе од иста боја. Промената на местата на елементите се врши преку функцијата *handle_swap()* и истовремено во неа се повикуваат и други функции кои проверуваат дали со одредена замена се случило совпаѓање и сл.

Откако ќе се отстранат некои елементи кои биле дел од совпаѓање, на нивното место мора да се појават нови елементи, па сето тоа се врши со *fill_empty_spaces()* функцијата која ги воочува празните места и на нивна позиција додава некој нов елемент.

Сите потребни исцртувања и пресметки во однос на промените во позиции се извршуваат во неколку функции и потоа тие се повикуваат при потреба. Тие се: *draw_all(), draw_grid(), is_adjacent(), swap_tiles(), animate_swap(), draw_score_level_and_moves().*

Едно ниво да се означи како комплетирано и да може успешно да се премине кон следното ниво се проверува дали играчот го постигнал бараниот резултат и во однос на тоа и на метрики како потези и време преку AI модулот се спрема новото ниво во *check_level_completed()* функцијата, која на крај го прикажува екранот за комплетирано ниво, односно *display_level_complete()* и исто така се прави ресетирање на целиот grid.

Во случај играчот неуспешно да заврши некое ниво, на екранот ќе му се појави game over исцртувања од *display_game_over()* функцијата и неговиот дотогашен најдобар резултат ќе се зачува како highscore во save_data.json фајлот.

- **Главниот game loop**
  Овој дел од кодот се грижи се додека играта е пуштена се да оди по својот редослед, односно го вчитува времето, додава бомби доколку е потребно и ги исцртува потребните екрани во зависност кое копче го кликнал играчот.