Dokumentacja postępów prac projektowych

Rozproszona platforma do tworzenia gier planszowych, przeznaczona na system Android.

*Niniejsze opracowanie powstało w trakcie i jako rezultat za­jęć dy­dak­tycz­nych z przedmiotu wymienionego na stronie tytułowej, pro­wa­dzo­nych w Akademii Górniczo-Hutniczej w Krakowie (AGH) przez oso­bę (oso­by) wy­mie­nioną (wymienione) po słowach "Pro­wa­dzą­cy zajęcia" i nie może być wy­korzystywane w jakikolwiek sposób i do jakichkolwiek ce­lów, w ca­ło­ści lub części, w szczególności pub­li­ko­wa­ne w ja­ki­kol­wiek spo­sób i w jakiejkolwiek formie, bez uzy­ska­nia uprzed­niej, pi­sem­nej zgody tej oso­by (tych osób) lub odpowiednich władz AGH.****Copyright © 2013 Akademia Górniczo-Hutnicza (AGH) w Krakowie***

**Spis treści**

1. Zakres dokumentu 3

1.1. Postępy prac projektowych 3

1.2. Opis postępów prac 4

2. Podsumowanie 6

3. Bibliografia 7

Indeks ważniejszych terminów 8

Spis ilustracji i tabel 8

# Zakres dokumentu

Dokument niniejszy zawiera informację jaki był postęp prac projektowych na kolejnych etapach jego implementacji, jakie problemy napotkaliśmy w trakcie i podsumowanie sposobu prowadzenia prac projektowych który obraliśmy.

# Postępy prac projektowych

Najlepszą charakterystyką prac projektowych będzie niniejszy wykres:

Stan prac jest subiektywną oceną postępów mierzona w skali 0- rozpoczęcie , 9 – gotowy prototyp platformy i przykładowy prototyp gry Monopoly. Dzień zgodnie z nazwą określa ilość dni od rozpoczęcia prac nad projektem.

Oto tabelka z podsumowaniem wykresu:

|  |  |  |
| --- | --- | --- |
| Dzień | Stan Prac | Opis |
| 0 | 0 | Rozpoczęcie prac projektowych |
| 7 | 1 | Zebranie wymagań |
| 17 | 2 | Tworzenie wysokopioziomowej architektury rozwiązania |
| 31 | 3 | Tworzenie diagramu klas |
| 45 | 4 | Tworzenie prototypu rdzenia platformy |
| 52 | 5 | Rozpoczęcie pisania prototypu gry Monopoly |
| 59 | 6 | Dalsza prace nad prototypem gry Monopoly |
| 64 | 2 | Ponowne rozpatrzenie wysokopoziomowej architektury rozwiązania |
| 66 | 3 | Stworzenie wysokopoziomowej architektury rozwiązania |
| 69 | 4 | Pisanie klas w pomocniczych i testów |
| 72 | 5 | Rozszerzanie rdzenia rozwiązania przy tworzeniu jednoczesnych testów. |
| 74 | 6 | Rozpoczęcie pisania prototypu gry Monopoly |
| 78 | 7 | Dalsze prace nad prototypem gry Monopoly przy ciągłym rozszerzaniu rdzenia rozwiązania i testach |
| 82 | 9 | Gotowy prototyp gry i rdzenia |

## Opis postępów prac

Mimo trudnego początku jakim było zebranie wymagań w stosunku do projektu jak i naszego początkowego zbytniego upraszczania złożoności projektu, prace projektowe rozpoczęliśmy w miarę zgodnie z harmonogramem który początkowo sobie narzuciliśmy. W stosunkowo krótkim czasie powstała koncepcja rdzenia rozwiązania która nie uległa dużym zmianom do końca jeśli chodzi o wysokopoziomową myśl która jej przyświecała. Po rozpoznaniu i wstępnym diagramie klas jak i architektury rozpoczęliśmy implementację prototypu platformy wraz z prototypem gry Monopoly.

Lecz, co widać na załączonym wykresie, w trakcie implementacji prototypu nastąpił powrót do jednego z etapów który został teoretycznie wcześniej zamknięty. Decyzja powrotu do tworzenia architektury wysokopoziomowej rozwiązania została przez nas podjęta gdy tworzenie prototypu gry Monopoly na platformie napotykała na niespotykane trudności. W trakcie tworzenia prototypu Monopoly w wyniku braku testów rdzenia samej platformy stale napotykaliśmy się na błędy. W tym samym czasie brak mechanizmów wspomagających twórcę gry powodował , że zrobienie najprostszych i rutynowych ( dla gier planszowych ) funkcjonalności jak przesunięcie pionka po planszy , czy stworzenie najprostszej planszy z początkiem i końcem które są dla siebie sąsiadami nastręczało za dużych trudności.

Kolejna rzeczą wartą powiedzenia przez którą zweryfikowaliśmy i zmieniliśmy architekturę platformy był brak wystarczającej elastyczności pomiędzy turą gracza, pobraniem ruchu i pokazaniem wyników. Przy poprzednim projekcie architektury kolejność tych etapów była wymuszona przez platformę co czyniło bardzo trudnym ( momentami niemożliwym) aby móc zsynchronizować pobranie tury od gracza zarazem stale pokazując mu efekty jego działań.

Kolejna wadą rozwiązania poprzedniego było ciche założenie , że wszyscy gracze powinni widzieć to samo na mapie, co w przypadkach strategicznych gier planszowych się totalnie nie sprawdza , gdyż każdy z graczy mógłby mieć odsłoniętą pewną część mapy ale nie widzieć jej całości czy też schowanych wojsk rywala.

W architekturze poprzedniego rozwiązania kolejną sprawą która okazała się rodzić problemy była implementacja logiki turowej , która znajdowała się w metodzie obiektu przekazywanego nam przez gracza. W praktyce podczas implementacji natknęliśmy się na niespodziewany duży rozrost tej metody, ogromną liczbę instrukcji IF i bardzo trudny do zrozumienia i implementacji kod. Architektura została w związku z tym przeanalizowana i zrefaktoryzowana.

Mając doświadczenia z poprzedniej nieudanej próby zrobienia prototypu m.in. z powodu błędów samej platformy przyjęliśmy model przyrostowy tworzenia platformy. Rozpoczęliśmy jej budowę od najmniejszych składników jakimi były pionki , plansza , przedmioty, testując od samego początku wszystkie jej składniki. W każdym kolejnym etapie dodawaliśmy kolejne funkcjonalności razem z testami i podejmując decyzje architektoniczne na każdym etapie uwzględniając nasze wcześniejsze doświadczenia.

Z doświadczeniem jakie zdobyliśmy na nieudanej implementacji Monopoly, po ustanowieniu nowej architektury i przeprowadzaniu testów od samego początku postępy prac nad projektem rosły w tempie nieporównywalnym do naszych wcześniejszych doświadczeń. Programowanie i projektowanie przyrostowe w ramach tego projektu okazało się , po ustanowieniu wstępnej architektury, bardzo pasujące do wykonywanego projektu biorąc pod uwagę zarówno kwestię architektury jak i implementacji .

# Podsumowanie

Wydaje się , że najważniejszą konkluzją z prowadzenia prac projektowych jest to, że ustanowienie dobrego designu i API wymaga czasu i stałych prób. Pierwszy stworzony design jak by nie wydawał się właściwy ,z dużą dozą prawdopodobieństwa okaże się niewystarczający i zbyt trudny w praktyce do zastosowania. Zadanie ustanowienia odpowiedniej architektury „ na raz” wydaje się nam z perspektywy zdobytych doświadczeń wręcz niemożliwe.

Następną rzeczą która okazała się niezwykle istotna było przeprowadzanie testów jednostkowych. Po pierwsze wyeliminowała błędy platformy które przy nieudanej próbie zrobienia prototypu gry Monopoly wyszły , a po drugie wpływała niespodziewanie na API poszczególnych składników. Przy przeprowadzaniu testów małej funkcjonalności stale dopisywaliśmy metody w odpowiednich obiektach gdzie zastosowanie danej rzeczy było trudne , co później okazało się zbawienne w trakcie pisania samego prototypu Monopoly.

Decyzja o powrocie stanu prac projektu do ponownego przemyślenia i zmian w architekturze okazała się słuszna w perspektywie późniejszych prac. Można stwierdzić , że sztywne trzymanie się wcześniejszych decyzji architektonicznych i projektu klas prawdopodobnie przyniosłoby klęskę całemu projektowi który nawet przy zrealizowaniu założeń funkcjonalnych , byłby w praktyce tak trudny do nauki i wykorzystania ,że jego wykorzystanie nie miałoby żadnego sensu w rzeczywistych projektach gier.

Elastyczność i luźne powiązania między klasami sprawiły , że zajmując się danym zagadnieniem , nie byliśmy zmuszeni do rozmyślań o pozostałych częściach które w danym momencie były juz przetestowane. Realizacja gry przez sieć wymagała dopisania funkcjonalności do modelu gry lokalnej nie zmieniając w praktycznie żadnym stopniu wcześniejszego kodu . W trakcie realizacji platformy szereg wzorców projektowych znacznie ułatwił nam osiągnięcie elastyczności kodu i ułatwienia z jej korzystania.

Dla przykładu model rozgrywki gry planszowej oparty na wzorcu Stan wydaje się idealnie pasować do jego charakterystyki i znacznie ułatwia pisanie tego modelu przez twórcę gry na platformie. W naszej subiektywnej opinii korzystanie z platformy przez twórcę gry zostało uproszczone w takim stopniu ,że po niewielkim zapoznaniu się z dokumentacją użytkownika i Javadokiem jest on w stanie rozpocząć swój projekt.

# Bibliografia

1. Brett D. McLaughlin, Gary Police & David West: *Object Oriented Analysis and Design*, O’Reilly, 2007

Indeks ważniejszych terminów

**Nie można odnaleźć żadnych haseł indeksu.**

1. Spis ilustracji i tabel

Ilustracje

**Nie można odnaleźć pozycji dla spisu ilustracji.**

Tabele