Szósty zestaw zadań - Proces Testowy

Grupa: Artur Jankowski, Paweł Sikora

Gra: Europa Universalis 4

Planowanie i nadzór nad testami

Strategia testowania i podejście do testów

Wstępnie planujemy testować grę w sposób dynamiczny i heurystyczny, tzn. jako że gra jest mocno rozbudowana i wiele mechanik wpływa na siebie w znaczącym stopniu, chcemy dać testerom wolną rękę przy testowaniu, tak aby mogli oni wykryć jak najwięcej nieszablonowych błędów.

Dla najważniejszych mechanik i produkujących najwięcej błędów oraz najprostszych funkcjonalności będziemy stosować podejście regresywne, czyli takie w którym chcemy zautomatyzować testy funkcjonalne.

Cele testów

- wykrycie frustrujących gracza błędów
- spełnienie wymagań niefunkcjonalnych
- przetestowanie testów dotyczących historyjek użytkownika
- możliwość prezentacji częściowo ukończonego produktu
- przyśpiesznie produkcji, dzięki ominięciu ponownego naprawiania tych samych błędów

Metryki

- liczba zgłoszonych/otwartch/naprawionych błędów
- ankieta skierowana do graczy na etapie prototypowania i na kolejnych interacjach interfejsu.
- całkowita liczba testów z wyróżnieniem na moduły
- liczba testów zautomatyzowanych
- stopień pokrycia kodu, warunków, rozgałęzień, decyzji.
- liczba pokrytych przypadków użycia

Analiza testów

Podstawy testów

- wymagania systemowe gry
- specyfikacja mechanik gry

- specyfikacja user infertace gry
- specyfikacja wymagań funkcjonalnych i niefunkcjonalnych

Podstawowe warunki testowe

- wydzielenie testowanej mechaniki bez wpływu innych nie-testowanych
- poprawnie działające według założeń mechaniki

Historyjki

1. Jako gracz, chciałbym zrekrutować nowe jednostki do mojej armii, żeby była ona gotowa do stoczenia bitwy.
2. Jako gracz, chciałbym wybudować nowe budynki w moich miastach, aby bardziej je rozwinąć.
3. Jako gracz, chciałbym wysłać handlarza do punktu handlowego, aby zwiększyć tam mój wpływ i zarobki.

Historyjki użytkownika spełniają zasadę INVEST oraz są jednoznaczne.

Wymagania funkcjonalne

- gracz może rekrutować nowe jednostki do armii
- gracz może stoczyć bitwę wchodząc w interkację z wrogimi jednostkami swoją armią
- gracz powinien móc wybudować nowe budynki w miastach
- gracz, wchodząc w interakcję z punktem handlowym, może wysłać handlarza w dane miejsce
- AI wchodzi w interakcję z graczem i odpowiednio reaguje na jego poczynania
- rozwój naukowy wpływa na modyfikatory gracza i otwiera nowe funkcjonalności przed graczem

Wymagania niefunkcjonalne

- szybkość działania
- minimalne wymagania sprzętowe:
  - OS: XP/Vista/Windows 7/Windows 8
  - procesor: Intel Pentium IV 2.4 GHz lub AMD 3500+
  - pamięć: 2 GB RAM
  - karta graficzna: NVIDIA GeForce 8800 lub ATI Radeon X1900, 512mb VRAM
  - DirectX: 9.0c
  - dysk twardy: 2 GB wolnego miejsca
- użyteczność interfejsu

- poziom trudności powinien być odpowiednio dopasowany do opisu
- zoptymalizowane wykorzystanie pamięci RAM

Projektowanie testów

Przypadki testowe dla historyjek

Przypadek 1: ID: WF_1

Tytuł: Rekrutacja jednostek

Opis: Rekrutowanie nowych jednostek do armii.

Warunki wstępne: Wybrana prowincja musi być niepodległa.

Kroki:

1. Gracz wybiera prowincję.
2. System wyświetla okno prowincji.
3. Gracz klika na zakładkę werbowania.
4. System wyświetla okno werbowania.
5. Gracz wybiera jednostkę, którą chce zrekrutować.
6. System dodaje jednostkę do kolejki rekrutacji oraz system zabrał graczowi zasoby, które ona kosztowała.
7. Po upływie określonego dla typu jednostki czasu, na prowincji pojawia się jednostka.

Warunki końcowe: Gracz zrekrutował wybraną liczbę jednostek, które pojawią się na wybranej prowincji. Graczowi została zabrana odpowiednia liczba pieniędzy i zasobów ludzkich.

Przypadek 2: ID: WF_2

Tytuł: Stawianie budynków

Opis: Gracz stawia nowe budynki.

Warunki wstępne: Wybrana prowincja musi być niepodległa. Gracz ma wolny slot do budowania.

Kroki:

1. Gracz wybiera prowincję.
2. System wyświetla okno prowincji.
3. Gracz klika na zakładkę budowania.
4. System wyświetla okno budowania.

5. Gracz wybiera budynek, który chce wybudować.
6. System dodaje budynek do kolejki budowania oraz system zabrał graczowi zasoby, które ona kosztowała.
7. Po upływie określonego dla typu budynku czasu, w prowincji powstaje nowa budowla.
8. System aktualizuje modyfikatory bonusów dla miasta.

Warunki końcowe: W wybranej prowincji powstał nowy budynek. Odpowiednie zasoby zostały graczowi zabrane. Miasto zostało ulepszone, a modyfikatory bonusów zwiększyły się.

Przypadek 3: ID: WF_3

Tytuł: Wysłanie handlarza

Opis: Gracz wysyła handlarza (kupca) do wybranego regionu handlowego, za pomocą opcji "Collect from trade".

Warunki wstępne: Gracz posiada kupców.

Kroki:

1. Gracz wybiera prowincję, która zawiera region handlowy.
2. System wyświetla okno prowincji.
3. Gracz klika na zakładkę handlu.
4. System wyświetla okno regionu handlowego.
5. Gracz klika przycisk "Collect from trade"
6. System wysyła do wybranego regionu handlarza.
7. System aktualizuje dane dotyczące wpływu gracza w danym regionie, jego zarobków oraz Trade Power.

Warunki końcowe: Kupiec znajduje się w regionie handlowym. Trade Power regionu, wpływy gracza oraz zarobki zwiększyły się.

Przypadki testowe dla wymagań niefunkcjolanych

Przypadek 1: ID: WN_1

Tytuł: Zoptymalizowane wykorzystanie pamięci RAM

Opis: Przetestowanie zużycia pamięci RAM.

Warunki wstępne: Ilość pamięci RAM zainstalowana na komputerze równa minimalnym wymaganiom.

Kroki:

1. Uruchomienie gry na najniższych ustawieniach.
2. Wykonywanie wszelkich obciążających komputer czynności w grze.
3. Monitorowanie na bieżąco zużycia pamięci RAM.

Warunki końcowe: Gra wykorzystała nie więcej pamięci RAM, niż 2 GB.

Przypadek 2: ID: WN_2

Tytuł: Minimalne wymagania sprzętowe

Opis: Przetestowanie działania minimalnych wymagań sprzętowych.

Warunki wstępne: Sprzęt komputerowy z podzespołami podanymi w wymaganiach.

Kroki:

1. Uruchomienie gry na najniższych ustawieniach.
2. Wykonywanie wszelkich obciążających komputer czynności w grze.
3. Monitorowanie na bieżąco płynności gry.
4. Spisywanie funkcjonalności, przy których wystąpiły znaczne spadki FPS.

Warunki końcowe: Gra działała płynnie (minimalnie 30 FPS), nie wystąpiły żadne błędy związane z pamięcią (np. memory management error).

Przypadek 3: ID: WN_3

Tytuł: Poziom trudności powinien być odpowiednio dopasowany do opisu

Opis: Gracz przechodzi rozgrywkę na różnych poziomach trudności.

Warunki wstępne: -

Kroki:

1. Gracz wybiera poziom trudności.
2. Gracz uruchamia grę.
3. Gracz gra w sposób dowolny dla siebie.
4. Spisywanie rzeczy, które w odczuciu mogą być za proste/trudne na dany poziom trudności.

Warunki końcowe: Poziomy trudności możliwe do wyboru są odpowiednio dostosowane do swojej nazwy i opisu.

Implementacja testów

Tworzymy odpowiednie środowisko testowe, dane testowe, oraz otrzymujemy testowany kod. Gwarantujemy, że przypadki testowe zostały opisane i są gotowe do wykonania.

Powinno się wyspecyfikować, które z testów należy implementować wcześnie, a które dotyczą jeszcze zmieniających się obszarów gry. Szczególnie ważne jest to przy testach automatycznych.

Przygotowane środowisko i dane należy opisać, co ułatwi przyszłe testowanie regresywne.

Należy utworzyć skrypty automatyzujące proste lub często powtarzane testy.

Wykonanie testów

Sprawdzamy, czy gotowe i prawidłowo skonfigurowane jest środowisko testowe oraz narzędzia. Sprwadzamy też, zy otrzymaliśmy prawidłową wersję kodu od programistów oraz dane testowe od testerów.

Dane dotyczące rezultatów oraz wynikające metryki podczas testowania powinny być śledzone, gromadzone, przesyłane oraz analizowane.

Testy wykonywane na podstawie przypadków testowych zostawiają testerowi pewne pole dowolności, tak by wykryć nieszablonowe błędy.

Wszystkie znalezione błędy powinny być odpowiednio spisane w raporcie, który powinien zawierać m.in.: dane testera, datę wykonania testu, sprzęt komputerowy, opis błędu, lista kroków potrzebna do wywołania błędu.

Ocena kryteriów zakończenia i raportowanie

Do oceny kryteriów zakończenia należy spełnienie metryk dla wymagań. Przykładowo:

- ocena wynikająca z ankiet musi wynosić powyżej 8.5 w 10 punktowej skali,
- należy przetestować 100% kluczowych mechanik,
- pokrycie kodu powinno wynieść powyżej 80%,
- przejście testów akceptacyjnych.

Ważna jest też ocena klienta oraz opinia Kierownika Testów co do wiarygodności informacji.

Raportowanie i interpretacja metryk powinna pokazywać zmiany metryki w czasie, tak by można było ocenić zmiany, i próbować przewidzieć trendy, oraz przewidywać koszty względem możliwych zysków.

Należy raportować dane dotyczące pokrycia kodu, ryzyka dotyczącego dalej istniejących błędów, szacowanej miary staranności oraz kosztu procesu testowania.

Czynności zamykające test

Należy utworzyć raport dotyczący ukończenia testowania. Należy opisać w nim wykonane testy, wskazane błędy, pominięte testy wraz z uzasadnieniem pominięcia.

Należy określić błędy lub usterki, których naprawienie zostaje odroczone, lub jest niemożliwe wraz z uzasadnieniem decyzji.

Błędy, wraz z raportami wykryte w trakcie testowania należy przekazać deweloperem do naprawienia i ponownego testowania, mając na uwadze, że błędy w grach pojawiają się nawet lata po ich dacie premiery. Wszystkie dane, raporty należy poddać archiwizacji.

Ostatni etap to wyciągnięcie wniosków dotyczących procesu testowego, wykrycie niewłaściwych praktyk, utrwalenie dobrych, planowanie przyszłych procesów. Należy zastanowić się jak można ulepszyć proces, oraz czy testowanie było skuteczne