Jak tworzyć własne gry?

Czyli krótki poradnik „jak zostać programistą gier”.

Slajd 1 – slajd tytułowy

Kto z Was kiedykolwiek zrobił jakąś własną grę? A chociaż spróbował? A fajne to jest? :)

Prezentacja – na temat [^]

Cel: żeby każdy wychodząc z tej sali wiedział co zrobić, żeby napisać fajną grę.

Slajd 2 – Zagadnienia

- Dlaczego programista gier?
- Czego potrzebuje programista gier?
- Jak napisać i wydać grę?
- Czy jest jakieś środowisko programistów gier?

Slajd 3 – Dlaczego zostać programistą gier?

- Zajęcie kreatywne
- W dobre gry grają dziesiątki milionów ludzi
- Łączy różne dziedziny – matematykę, kinematykę, teorię grafów, sieci neuronowe, grafikę komputerową, przekaz medialny, literaturę... itd.
- Wszyscy lubimy gry, nieprawdaż? :)

Slajd 4 – Czego potrzebuje programista gier?

<twarde>

- Znajomość programowania (język dowolny, ale C++ najpopularniejszy; niektóre języki mogą przydać się jako skryptowe)
    - Assembler już tak bardzo nie przydaje się do optymalizacji ;)
- Matematyka <<niestety>>, zwłaszcza podstawy algebry 3D
    - Operacje na wektorach, macierzach
- Fizyka – poziom licealny starcza na dobry początek
    - Równania ruchu
    - Bryła sztywna (bardziej zaawansowane, ale bardziej realistyczne)

Slajd 5 – Czego potrzebuje programista gier?

<miękkie>

- Wytrwałość
  - Pisanie gry to miesiące, a czasem lata pracy
    projekt nie raz się zarzuca
- Umiejętność pracy zespołowej
  - Większych gier nie robi się samemu, nawet do tych niedużych potrzeba przynajmniej kogoś od grafiki
- Kreatywność
  - Pisanie gier to sztuka, liczą się ciekawe i pomysłowe rozwiązania
  - Pisanie gier to rozwiązywanie problemów, wymaga kreatywnego i często nieszablonowego podejścia do programowania
  - Czytałem o grze w okręty napisanej w SQL :). Grałem w tekstowe Hack&Slash napisane w Bashu. Grę można zrobić ze wszystkiego.

Slajd 6 – Jakich narzędzi potrzebuje?

- Kompilator / Środowisko programistyczne
  - Visual C++ - jedyny liczący się na Windowsie; wersje Express (2k5 i 2k8) mają licencję na aplikacje komercyjne
  - Code::Blocks – możliwość podpinania różnych kompilatorów, w tym świetne wsparcie dla kompiltora VC++
- Papier i ołówek / długopis – najlepszy przyjaciel programisty gier!
- Czajnik elektryczny – programowanie gier wymaga sporej ilości kawy lub herbaty

Slajd 7 – Wiedza

- Książki
  - Do języka
  - Do bibliotek graficznych – „OpenGL Programowanie” Gier, „DirectX Rendering w Czasie Rzeczywistym”
  - O programowaniu gier ogólne – GPG1,2,3,... 6 ;)
- Internet – artykuły (strony takie jak gamedev.net, gamedev.pl i .net)

Slajd 8 – Pomysły

Programowanie gier – kreatywne zajęcie; potrzeba wiele pomysłów.

Jak mieć ich dużo?

- Notować wszystko!
- Dużo grać – można coś podpatrzeć
- Dużo czytać – nie tylko programistycznych rzeczy, ale też książek technicznych i literatury pięknej
- Być świetnym obserwatorem – zarówno gier jak i świata rzeczywistego

Slajd 9 – Jak napisać własną grę – poradnik <<ta najbardziej interesująca część>>

Krok po kroku:

- Pomysł

- Projekt
- Dokumentacja
- Czas zacząć szukać sobie grafika :)
- Kodowanie
- Testowanie (tu strzałeczka do kodowanie)
- Wydanie
- Koniec

Proste, czyż nie ? :)

Slajd 10 – Z czego składa się gra?

Silnik – taki wewnętrzny system oprecyjny, udostępnia różne rzeczy:

- Obsługa błędów – logger, komunikaty diagnostyczne, reagowanie na błędy takie jak brak pliku z teksturą
- Wejście – obsługa klawiatury, myszy, joystick'a, itp.
- Dźwięk – odtwarzanie dźwięków, muzyki
- Grafika – renderer – w dużych grach bardzo skomplikowana część - wyświetlanie na ekranie obiektów gry

Slajd 11 – Pomysł i projekt

Mamy pomysł – co z nim robimy?

- Emocjonalne podejście – musimy żyć projektem, myśleć o nim dużo; pomysł ma rozpalać naszą wyobraźnię
- Burza mózgów – wypisanie wszystkich rzeczy, które nam przychodzą do głowy – np. na dużej kartce papieru (lub wielu). Także szkice i rysunki poglądowe.
- Uporządkowanie tego, co sobie wypisaliśmy, przyda się do pisania dokumentacji

Slajd 12 – Dokumentacja

- Zanim powstanie pierwsza linia kodu, musi istnieć przynajmniej ogólna dokumentacja techniczna i dokładna dokumentacja projektowa! „Należy dwa razy pomyśleć, nim zacznie się programować, aby nie trzeba było dwa razy programować, nim zacznie się myśleć”
- Design Document
    - O grze – od strony gracza
    - Szkice, rysunki poglądowe, opis działania menu, jak gracz steruje postacią, warunki wygranej, przegranej
    - Czyli generalnie spec taki jak dr Bułka tłumaczył :)
- Technical Document – szczegóły implementacyjne
    - Implementacja wejścia, grafiki – jakie interfejsy stworzymy, jak będzie przebiegać komunikacja między nimi
    - Szczegóły użytych algorytmów – jeśli je wymyślamy samemu to warto je sobie rozpisać!

Slajd 13 – Szukanie i zatrudnienie sobie grafika ;)

- Grafiki jest więcej niż się na początku wydaje, nie każdy ma zdolności graficzne

- Pracować z grafikiem jest weselej ;).
- Dokumentacja pozwala stwierdzić, jaka grafika jest potrzebna, ustanawia warunki jego pracy
- Grafika trzeba znaleźć i zachęcić do współpracy...
- ...a często też mu zapłacić ;))

Slajd 14 – Kodowanie

- Podział gry na moduły – tak jak te opisane wyżej
- Zasada: najpierw projektuj, potem klep – ten cytat co był przy dokumentacji. Jeśli czujesz, że nie wiesz co w danej chwili masz napisać, to wyłącz kompa, wróć do dokumentacji i papieru.
- Dobry styl – bardzo ważny, bo często sami używamy swojego kodu po latach
- Zasady poważnego programowania (dla wychowanych na algorytmice)
    - Efektywne != najszybsze – słowo 'efektywność' oznacza nie tylko szybkość, ale też i to jak dobrze spełnia swoje zadanie i jak łatwo jest taki kod poprawiać
    - Quality > Speed – liczy się przede wszystkim bezbłędność i błędoodporność kodu, jego jakość i możliwość ponownego wykorzystania, oraz rozmiar (im mniej tym lepiej)

Slajd 15 – Testowanie

- Testujemy na bieżąco w trakcie klepania – kompilator nam pomaga
    - Unit testy (w C++ CppUnit) – sprawdzanie każdego modułu osobno
- Wersja Alpha i Beta
- Dobry i łatwy kontakt z graczami – naprawianie błędów już po wydaniu

Slajd 16 – Wydanie

- Przygotowanie materiałów – screeny, opisy
- Zrobienie ładnej strony WWW
- Poszukanie wydawcy
    - Udostępnić na stronie
    - Upchnąć na cover-cd jakiegoś czasopisma
    - Skontaktować się z wydawcami (np. Play.PL wydaje proste gry)
- Dobra gra wydaje się sama! Wydawcy czasem kupują gry (np. Krajew4 – FruitLogic), a i szukają dobrych programistów do zespołów

Slajdy 17,18 – Społeczności programistów

- Za granicą
    - Gamedev.Net
        - Bardzo dużo artykułów, dobre forum społeczności
    - GamaSutra
        - Dużo artykułów, pisanych przez ludzi ze znanych i poważanych firm w branży
        - Oferty pracy

- W kraju
  - Gamedev.PL
    - Słynny Warsztat – forum, dużo artykułów, kanał IRC
    - Społeczność
    - Kolega Bogumił na przykład ;) :)
    - Dużo projektów amatorskich – mało skończonych :)
  - Compo – konkurs na gry organizowany przez warsztatowiczów
    - Różne edycje – one week, one month, 64k – zawsze coś ciekawego
    - Warsztat Summer of Code 2007 w te wakacje; powstały gry o jakości komercyjnej, w tym wspomniany FruitLogic
  - IGK
    - W Siedlcach, teraz to 5-ta będzie :), na początku kwietnia
    - Inicjatywa Warsztatowiczów
    - 3 dni; wykłady, Compo
    - Obecni ludzie z branży

Slajd 19 – Materiały dodatkowe

- Pytania?

Slajd 20 – Credits and thanks

KTHXBYE

Jacek Złydach

temporal.pl@gmail.com

AGH, EAIE, Informatyka Stosowana 2007+

NOTES:

Prelegent gra sobie we Fruit Logic przed rozpoczęciem prezentacji ;) //jeśli istnieje taka możliwość.