Programowanie gier w PyGame

Łukasz Milewski

Uniwersytet Wrocławski

April 20, 2011, Wrocław

Target

Dla kogo jest ten wykład?

- Początkujący programiści
- Osoby chcę tworzyć gry dla zabawy
- Osoby oczekujące 'szybkich' efektów
- Chętni do uczenia się nowych rzeczy

Dla kogo to nie jest wykład?

- Nastawieni na komercyjny gamedev
- Zaawansowani programiści
- Osoby chcące tworzyć gry na konsole

OOP, DOD, CBP i inne literki

Gra amatorska to nie komercyjne oprogramowanie

Czego nie warto robić

- OOP, DOD, CBP, TDD i inne literki
- Clean code
- Pisanie własnego frameworka, enginu, czegokolwiek
- Uogólnianie

Co warto robić

- Pisać grę
- Dodawać kolejne featury
- Stosować KISS
- Maksimum efektów przy minimalnym nakładzie pracy

System kontroli wersji

github

- https://github.com/
- http://help.github.com/create-a-repo/

git

- git clone
- git push origin master - za pierwszym razem
- git commit -a -m 'tutaj opis zmian'
- git push
- gitk

System kontroli wersji

git a paraca grupowa

- git remote add NAZWA ADRES
- git fetch NAZWA
- git merge NAZWA/master (integrator)
- git rebase NAZWA/master (collaborator)

Motywacja

- Programowanie z kimś jeszcze (jak wybrać taką osobę?)
- Konkursy (np. www.pyweek.org, compo)

Dwie najskuteczniejsze metody nauczenia się progrmowania gier

Pisz gry

Praktyka czyni mistrza. Co więcej - każda, nawet najprostsza, skończona gra daje sporą dawkę motywacji do tworzenia kolejnych. Najwięcej można się nauczyć rozwiązując konkretne problemy z konkretną grą. Uczenie się z tutoriali (lub co gorsze - z książek) jest bardzo nieefektywne.

Czytaj kod

Na www.pyweek.org oraz www.pygame.org jest bardzo wiele przykładowych gier. Warto wybierać np. te, które wygrywały poprzednie edycje pyweeka i zobaczyć jak są zrobione. Kod zazwyczaj nie jest piękny, obiektowy czy zgodny z inną ideologią. Za to działa, jest skończony, rozwiązuje konkretne problemy i ma więcej featurów niż pozostałe gry w danej edycji.

Dlaczego warto wybrać Python

Zalety

- Pygame
- Bardzo łatwy do nauczenia się
- REPL (toplevel)
- Można zrobić kompletną grę w tydzień
- Dużo bibliotek
- Bogata biblioteka standardowa (np. moduł random)

Wady Pythona

Wady

- Wydajność
- Wydajność
- Wydajność

Python - składnia

liczby, booleans, słowniki, krotki, if, listy, for, range

Zobacz data_types.py

slicing, list comprehensions

Zobacz lists.py

funkcje, klasy, metody

Zobaczk fun_class.py

Python - debugging

Jak wyszukiwać błędy?

- użyj debuggera [restart, p, c, b] (prezentacja)
- print debugging (moduł pprint)
- bisekcja

Python - moduły

sys

- import sys
- sys.argv (sound == not "–nosound" in sys.argv)
- sys.argv[0]

os

- import os
- base_path = os.path.abspath(os.path.dirname(sys.argv[0]))
- os.path.* (join, abspath, dirname, isfile)