ZPR - FIRST PERSON TEACHER

DOKUMENTACJA PROJEKTU

HTTP://CODE.GOOGLE.COM/P/FPTEACHER/

TEMAT: DOKUMENTACJA KOŃCOWA
AUTOR: SEBASTIAN ŁUCZAK. RAFAŁ MALINOWSKI, CEZARY ZAWADKA
DATA EDYCJI: 12-01-2010
STATUS: FINALNY

SPIS TREŚCI

WSTĘP

Realizowany projekt miał być grą odzwierciedlającą przebieg wykładu z przymrużeniem oka.
W obecnej chwili FPTeacher jest aplikacją demonstrującą możliwości silnika FPTEngine będącego szkieletem stworzonym na potrzeby wspomnianej gry.

FPTeacher, CZYLI DEMONSTRACJA MOŻLIWOŚCI FPTEngine

W SKRÓCIE O FPTEngine	FPTEngine jest multiplatformowym silnikiem gry 2D opartym o biblioteki SimpleDirectMedia Library i OpenGL. Przy jego implementacji stosowane były biblioteki z rodziny Boost. Do jego funkcjonalności należy akcelerowane sprzętowo wyświetlanie grafiki 2D w trybie okienkowym lub pełnoekranowym, pobieranie zdarzeń z klawiatury i myszy, odtwarzanie dźwięku lokalizowanego w przestrzeni, pobieranie danych z plików XML, zarządzanie zasobami (grafiki, animacje, dźwięki), komunikacja sieciowa z drugim terminalem w dwuosobowym trybie gry, synchronizacja i integracja wszystkich wcześniej wymienionych podsystemów. W dalszej części dokumentacji opisane zostaną wszystkie funkcjonalności z podziałem na klasy

NARZĘDZIA STOSOWANE PRZY IMPLEMENTACJI

MICROSOFT VISUAL STUDIO 2008 PROFESSIONAL EDITION + ANKHSVN	Rozwój całej aplikacji, wspomaganie pracy zespołowej przez integrację obsługi repozytorium SVN z IDE
VISUAL LEAK DETECTOR 1.9H	Darmowy detektor wycieków pamięci do MVS.
KDEVELOP 3.5.4	Budowa projektu na systemie operacyjnym opartym o Linux (Fedora 12)
CODE.GOOGLE.COM	Hosting projektu i repozytorium
SCRUMNINJA.COM	Narzędzie wspomagające prowadzenie projektu w oparciu o metodologię SCRUM, darmowe przy otwartych projektach
AUTODESK 3D STUDIO MAX 2009	Rendering i modelowanie sceny auli, darmowy dla zarejestrowanych studentów
GIMP 2.6	Tworzenie sekwencji animacji i drobnych elementów GUI
DOXYGEN 1.6.1	Dynamiczne tworzenie dokumentacji technicznej

ZASTOSOWANE BIBLIOTEKI

SimpleDirectMedia Library	Dostarcza międzyplatformową obsługę zdarzeń pochodzących z klawiatury i myszy, komunikację z systemem operacyjnym, i łatwą integrację z OpenGL oraz inne funkcjonalności przydatne w programowaniu gier, w tym timer'y. Dodatkowo wykorzystano rozszerzenia SDL: ➢ SDL_image - obsługa plików graficznych o różnorodnych formatach ➢ SDL_mixer - obsługa dźwięku ➢ SDL_net - dostarcza możliwość oprogramowania gniazdek sieciowych ➢ SDL_ttf - obsługa czcionek TrueType
OpenGL	Wyświetlanie grafiki akcelerowane sprzętowo.
BOOST	Pakiet dostarcza dodatkowych rozwiązań, brakujących w standardzie języka C++. Wykorzystano: boost::any, boost::archive, boost::serialization, boost::foreach, boost::multi_array, boost::smart_ptr, boost::thread, boost::bind
VISUAL LEAK DETECTOR	Detekcja wycieków w build'ach "Debug" w systemie Microsoft Windows

WYMAGANIA PRZY KOMPILACJI

MICROSOFT WINDOWS	Wszystkie wymienione powyżej biblioteki
LINUX	Wszystkie wymienione powyżej biblioteki, za wyjątkiem Visual Leak Detector W środowisku Linux aplikacja musi być uruchamiana z konsoli systemowej

OPIS DOSTARCZONEJ APLIKACJI

FPTeacher - FUNKCJONALNOŚĆ APLIKACJI DEMONSTRACYJNEJ

OGÓLNE	Aplikacja demonstracyjna pozwala na "rozgrywkę" w trybie jednoosobowym, bądź dwuosobowym, zależnie od wyboru. Aplikacja pozwala na uruchomienie pełnoekranowe, jednak z racji posiadania jednego zestawu grafik i stratnego skalowania obrazu na monitorach LCD, tryb ten nie jest prezentowany. GUI umieszczone na górze ekranu pozwala zatrzymać, zapauzować lub włączyć odgłosy tła, oraz opuścić aplikację.
TRYB JEDNOOSOBOWY	W trybie jednoosobowym prezentowana jest aula wypełniona przez studentów na podstawie pliku XML. Każdy student jest reprezentowany graficznie i wykonuje zestawy animacji siedzenia dobierane na podstawie ważonego losowania. Kliknięcie studenta powoduje odtworzenie krótkiej wypowiedzi, zlokalizowanej przestrzennie na sali zależnie od wartości x, y i wirtualnej z. Klikając na puste "miejsca" na auli można ją wypełniać losowo wybranymi studentami.
TRYB DWUOSOBOWY	W trybie dwuosobowym, po pomyślnym połączeniu (przez obu graczy wprowadzany jest adres IPv4, host wprowadza "0" co oznacza localhost) zależnie od roli (host lub gość) prezentowana jest aula od strony wykładowcy lub studentów. Interakcja z aulą jest taka sama, jednak jej efekty uwidaczniają się u obu "graczy" (odtwarzanie dźwięków rozlokowanych na sali i dodawanie studentów)

OPIS FUNKCJONALNOŚCI FPTEngine

FPTEngine można opisywać od różnych stron. Poniżej zawarty jest opis funkcjonalności oparty o strukturę klas. Dokumentacja techniczna została wygenerowana w doxygen.

CSingleton

Rola	Jest to szablon pozwalający w łatwy sposób, poprzez dziedziczenie, implementować wzorzec singletonu dla dowolnej klasy.
Zastosowanie	Dziedziczą po nim wszystkie klasy które są singletonami

CEntity

Rola	Abstrakcyjna klasa bazowa dla obiektów wyświetlanych na ekranie. Dostarcza interfejs z który dziedziczą klasy konkretne wykorzystywany przez CWorld.
Zastosowanie	Dziedziczą po niej CStaticEntity oraz CDinamicEntity

CWorld

Rola	Jest singletonem. Klasa zarządzająca obiektami dziedziczącymi po CEntity, między innymi sortuje je według współrzędnej z przygotowując do odrysowania zgodnie z algorytmem malarskim (najdalsze najpierw). Ta klasa wywołuje odrysowanie CEntity na ekranie. Każda CEntity po utworzeniu jest dodawana do kontenera w CWorld.
Zastosowanie	Używana przez CEngine do odrysowywania obiektów na ekranie.

CStaticEntity

Rola	Klasa obiektów wyświetlanych na ekranie które nie są animowane. W czasie rozgrywki elementy otoczenia są obiektami tej klasy.
Zastosowanie	W CWorld – elementy otoczenia

CDynamicEntity

Rola	Klasa obiektów wyświetlanych na ekranie które mogą być animowane. W czasie rozgrywki elementy studenci są obiektami tej klasy.
Zastosowanie	W CWorld – studenci

CField

Rola	Klasa opisuje jedno miejsce(siedzenie) na sali, ktore moze byc zajete przez studenta (CDynamicEntity, na potrzeby dema również CStaticEntity). Implementuje wzorzec obserwatora poprzez interfejs CMouseObserver. W zależności od stanu, po zinterpretowaniu CMouseEvent obiekt klasy generuje odpowiednie zdarzenia (odgrywanie kwestii studentow / dodawanie nowych studentow) oraz obiekty CNetworkEvent.
Zastosowanie	W klasie CAuditorium

CAuditorium

Rola	Jest singletonem. Klasa przedstawia salę, sklada sie z z pol CField (przechowywane są w kontenerze boost::multi_array), odpowiedzialna jest za za ladowanie sali na poczatku gry, wczytywanie konfiguracji z pliku konfiguracyjnego XML na a takze za pojawianie się nowych studentów na sali. Do wprowadzania danych konfiguracyjnych z pliku wykorzystywana jest biblioteka boost::serialization i archiwa XML z tej biblioteki.
Zastosowanie	Przy obsłudze sali wykładowej (zdarzenia, wyświetlanie itp)

CNetwork

Rola	Jest singletonem. Klasa odpowiedzialna jest za komunikacje sieciową. Nawiązuje połączenie TCP/IP z podanym hostem na podanym porcie (domyślnie 20010) a także zerializuje i odtwarza zdarzenia CNetworkEvent przez sieć. Wykorzystuje biblioteke SDL_net do komunikacji sieciowej a także boost::serialization do serializacji zdarzeń CNetworkEvent. Wysyłanie i odbieranie odbywa się w oddzielnych wątkach.
Zastosowanie	Przy komunikacji sieciowej

CNetworkEvent

Rola	Klasa bazowa dla zdarzeń przesłanych przez sieć. Dziedziczą po nie CSoundNetworkEvent orac CMouseNetworkEvent. Jest serializowana i przesyłana, a po drugiej stronie deserializowana i interpretowana w CNetwork. Obiekty pochodne tej klasy tworzone są przez obiekty CField po zinterpretowaniu CMouseEvent.
Zastosowanie	W CNetwork

CSoundNetworkEvent

Rola	Klasa dziedzicząca po CNetworkEvent pozwalająca przesłać informacje o dźwięku do odegrania.
Zastosowanie	W CNetwork, tworzona przez CField.

CStudentNetworkEvent

Rola	Klasa dziedzicząca po CNetworkEvent pozwalająca przesłać informacje o pojawieniu się nowego studenta na sali.
Zastosowanie	W CNetwork, tworzona przez CField.

CTimer

Rola	Jest singletonem. Klasa odpowiedzialna jest za odmierzanie czasu w grze. Wykorzystuje bibliotekę SDL_timer. Pozwala innym obiektom na śledzenie zegara (poprzez implementacje interfejsu CTimerObserver). Implementuje wzorzec adaptera dostarczając wygodnych metod zamiast funkcji biblioteki SDL_timer. Wykorzystywana do wywoływania cyklicznych zdarzeń na obserwatorach a także do usypiania wątków na określony czas.
Zastosowanie	W CNetwork, CEngine

CTimerObserver

Rola	Interfejs pozwalający na obserwację obiektu klasy CTimer. Obiekty tej klasy są powiadamiane o zdarzeniach cyklicznie. Możliwe jest śledzenie wielu interwałów jednocześnie.
Zastosowanie	CEngine implementuje ten interfejs

CHandle

Rola	Jest to szablon uchwytu, stosowany do dysponowania zasobami graficznymi i zestawami animacji. Do weryfikacji poprawności i niepowtarzalności uchwytu zastosowano magiczne liczby i unikatowe indeksy. Uchwyt każdego typu zasobu ma własny tag, aby zapewnić rozłączność typów zasobów.
Zastosowanie	W klasach zarządców zasobów i w klasach, które zabiegają o dostęp do zasobu. Szablon konkretyzowany jest w klasach zarządców do typu (typedef).

CHandleMgr

Rola	Jest to szablon zarządcy zasobami określonego typu DATA w oparciu o udostępnianie uchwytów. Zarządca przechowuje obiekty w kontenerze map indeksowane poprzez nazwy. Jeżeli przy zapytaniu o zasób, w bazie istnieje już instancja o danej nazwie, nie jest on dodawany, a zwracany jest po prostu do niego uchwyt.

	CHandleMgr ma niestety jedną wadę. Oparty na dereferencji uchwytu zwraca wskaźnik na obiekt wyłuskany z iteratora <vector>, co powoduje, że w razie rozrostu liczby zasobów ponad 1000 (zarezerwowane w konstruktorze) wskaźniki przestaną wskazywać poprawne wartości. Zostało to zaznaczone do poprawy w dalszej iteracji, ale nie zostało zaimplementowane.
Zastosowanie	W klasach konkretnych zarządców.

CSprite

Rola	Jest reprezentacją dwuwymiarowego zasobu graficznego. Ładuje się z pliku o podanej nazwie. Zależnie od tego, czy zasób jest zwykłą grafiką, czy sekwencją klatek, dokonuje odpowiedniego pocięcia i przygotowania do postaci tekstury OpenGL (używa do tego funkcji umieszczonej w utils.hpp). Dla powierzchni SDL we własnych metodach wykorzystuje sprytne wskaźniki shared_ptr z własnym dealokatorem. Początkowo CSprite nie posiadał zarządcy i w ten sposób bronił się przed wyciekami pamięci. CSprite mógłby być mniejszy, gdyby rola otwierania pliku i dołączania go do CSprite została przeniesiona do zarządcy.
Zastosowanie	Przechowywany przez zarządcę CSpriteMgr. Wyświetlany przez podsytem graficzny. Stosowany w zestawach animacji

CSpriteMgr

Rola	Jest to zarządca obiektów typu CSprite. Jest singletonem. Odpowiada za przechowywanie wszystkich zasobów graficznych w pojedynczych instancjach i udzielanie dostępu do nich pozostałym klasom. Korzysta z zarządcy opartego na uchwytach.
Zastosowanie	Używany przez wszystkie klasy potrzebujące dostępu do zasobów graficznych.

CAnimation

Rola	Jest reprezentacją zestawu animacji o określonej liczbie klatek, składającego się z wektora par uchwytów do obiektów typu CSprite i czasu odstępu między klatkami. Ładuje się z pliku o stosownej konstrukcji, który analizuje pod kątem obecności znaczników (charakterystycznych ciągów znaków) plik załadowany do strumienia. W sytuacji podania błędnej nazwy zestawu animacji, ładowany jest pojedynczy CSprite na podstawie obrazka wkompilowanego w kod.
Zastosowanie	Przechowywany przez zarządcę CAnimationMgr. Wykorzystywany przez obiekty CAnimator każdego CDynamicEntity.

CAnimationMgr

Rola	Jest to zarządca obiektów typu CAnimation. Jest singletonem. Odpowiada za przechowywanie wszystkich zestawów animacji w pojedynczych instancjach i udzielanie dostępu do nich pozostałym klasom. Korzysta z zarządcy opartego na uchwytach.
Zastosowanie	Używany przez wszystkie klasy potrzebujące dostępu do zestawów animacji. (obecnie CAnimator)

CAnimator

Rola	Jest podsystemem zarządzającym zestawami animacji w ramach danego CDynamicEntity. Komponuje z zestawów animacji sekwencje zgodnie z instrukcjami pozyskanymi z pliku sekwencji (nazwy i priorytety zestawów oraz tryb animacji). Zależnie od pobranych instrukcji, odtwarza sekwencje zapętlone, losowane zależnie od priorytetu akcji lub pojedynczą sekwencję. Przy błędnym pliku sekwencji ładuje pojedynczy CSprite na podstawie obrazka wkompilowanego w kod.
Zastosowanie	Używany przez CDynamicEntity do zarządzania jego sekwencjami animacji.

COGLWindow

Rola	Jest singletonem. Jego rola polega na inicjalizacji maszyny stanów OpenGL z odpowiednimi parametrami i stworzeniu okna o zadanej rozdzielczości, głębi barw i etykiecie, bądź wyświetleniu trybu pełnoekranowego o zadanych parametrach. W głównej pętli gry jest odpowiedzialny również za odświeżanie zawartości buforu karty graficznej.
Zastosowanie	Używany przez CEngine przy zarządzaniu oknem gry

CVideoSystem

Rola	Jest singletonem. Wbrew szumnej nazwie nie ma wielkiej roli, ta bowiem ogranicza się do wyświetlenia tekstury OpenGL przechowywanej przez wskazywany obiekt CSprite. Wyświetlanie nie ma parametru z, gdyż przy obsłudze przezroczystych tekstur wyłączony musi być Bufor Z. Silnik korzysta jednak z własnej współrzędnej z i sam sortuje CSprite'y pod jej kątem, by potem w odpowiedniej kolejności je wyświetlić (algorytm malarski), Potrzebne jest to, aby ławki zasłaniały postacie (postać to nie samo popiersie, a cała sylwetka, gdyż docelowo miała być możliwość przesiadania się po sali -> zmiany rzędów)
Zastosowanie	Używany przez CStaticEntity i CAnimator do odrysowywania grafiki na ekranie.

CSound

Rola	Klasa ta jest odpowiedzialna za przechowywanie informacji o dźwiękach. Posiada metody odpowiedzialne za włączenie, pauzowanie i wyłączanie dźwięków. Klasa ta wykorzystuje bibliotekę SDL_mixer. CSound ma jednak jedną wadę: nie ma możliwości odtwarzania jednocześnie tego samego dźwięki przez więcej niż jedno źródło.
Zastosowanie	Używany przez CAudioSystem jako obiekt przechowujący wszystkie informacje dotyczące dźwięku

CMusic

Rola	Klasa ta jest odpowiedzialna za przechowywanie informacji o muzyce. Posiada metody odpowiedzialne za włączenie, pauzowanie i wyłączanie muzyki. Klasa ta wykorzystuje bibliotekę SDL_mixer.
Zastosowanie	Używany przez CAudioSystem jako obiekt przechowujący wszystkie informacje dotyczące muzyki

CAudioSystem

Rola	Główna klasa dźwiękowa, która wykorzystując klasy CSound i CMusic odtwarza muzykę i dźwięki. Jest singletonem, przechowuje w sobie 2 zbiory, w jednym znajdują się wskaźniki na wszystkie muzyki, a w drugim wskaźniki na wszystkie dźwięki. Klasa ta przy użyciu odpowiednich metod zapewnia efekt dźwięku przestrzennego. Studenci siedzący w dalszych rzędach są słyszani gorzej i analogicznie ci siedzący w pierwszych rzędach są słyszani lepiej. Dodatkowo studenci siedzący na brzegach rzędów, są słyszani lepiej z lewego albo prawego głośnika.
Zastosowanie	Używany przez wiele klas do włączania i zatrzymywania wszystkich dźwięków i muzyki

CMouseEvent

Rola	Jest to klasa która jako swoje pola przechowuje informacje o tym w którym miejscu (współrzędne x,y) myszka została wciśnięta i w którym miejscu odciśnięta. Obiekty tej klasy są wysyłane jako argument metody refresh znajdującej się we wszystkich obserwatorach akcji związanych z myszką. Metoda refresh jest uruchamiana we wszystkich obserwatorach niezależnie czy kliknięcia dotyczą tych obserwatorów czy nie. Funkcjonowanie komunikacji między obserwowanym a obserwatorami można by usprawnić poprzez zmianę struktury klasy CMouseEvent, tak aby klasy których nie dotyczyło kliknięcie nie musiały analizować wszystkich pól CMouseEvent, tylko np. jedną flagę.
Zastosowanie	Używany jako argument metody refresh używanej przez obserwatorów akcji związanych z myszką

CMouseObserver

Rola	Jest to klasa abstrakcyjna po której dziedziczą wszystkie klasy zainteresowane zdarzeniami związanymi z myszą (CGui oraz CField). Ma ona czysto wirtualną metodę refresh, która jest wywoływana we wszystkich klasach dziedziczących za każdym razem gdy zażąda tego CInput.
Zastosowanie	Używany jako klasa po której dziedziczą wszyscy obserwatorzy myszki

CInput

Rola	Odpowiada za odbieranie sygnalow z myszy i klawiatury. Stan wciśniętych klawiszy przechowywany jest w tablicy, w której każdemu klawiszowi przypisana jest jedna komórka. W klasie tej znajduje się także mapa przechowująca wskaźniki do wszystkich obserwatorów, dziedziczących po klasie CMouseObserver, które są zainteresowane przychwytywaniem zdarzeń związanych z myszką. Do przesyłania tych informacji wykorzystano wzorzec obserwatora.
Zastosowanie	Używany do do przechwytywanie akcji związanych z klawiaturą i myszką

CGui

Rola	Klasa odpowiedzialna za wyświetlenie i funkcjonowanie przycisków związanych z Gui. Jest obserwatorem zdarzeń myszy, dziedziczy po klasie CMouseObserver. Przyciski są obiektami klasy CStaticEntity. Analizowanie kliknięć myszką w GUI jest rozwiązane nie do końca poprawnie, gdyż jest ustawione „na sztywno”. Klasę mogła by zostać ulepszona: powinna zostać napisana dodatkowa klasa lub metoda, która analizowałaby kliknięcia w gui w lepszy sposób, ponieważ obecnie zdarzenia związane z funkcjonowaniem Gui ustawione są „na sztywno”.
Zastosowanie	Używany do obsługi funkcjonalności gui

WYKORZYSTANE WZORCE

Adapter, Obserwator, Singleton, Fabryka Obiektów (CNetwork)

PODSUMOWANIE I WNIOSKI Z ROZWOJU PROJEKTU

FPTeacher miał być grą. W dokumentacji wstępnej 70% nacisku przyłożyliśmy do zawartości tej gry, a nie do zagadnień technicznych. Dosyć szybko, wraz z lekturą kolejnych opracowań tematu, zdecydowaliśmy się na budowę solidnego silnika do tej gry. Zagadnienie to okazało się trudno skalowalne przy naszym doświadczeniu z pisaniem gier. Efektem tej decyzji jest FPTEngine.
Zamiast wąsko wyspecjalizowanego kodu stworzyliśmy dosyć uniwersalny szkielet, który może nosić gry o różnorodnej tematyce i typie rozgrywki. Szkielet zgodny z konwencją izolacji logiki od danych, wielowątkowy, obsługujący komunikację sieciową między różnymi systemami operacyjnymi, wolny od wycieków pamięci i zbędnego duplikowania dużych obiektów (struktur noszących w sobie dane o grafice i dźwięku), kompletnie udokumentowany.
Dobrze się stało, że rozwijaliśmy właśnie ten, a nie inny projekt. Dzięki ogromowi pracy jaki musieliśmy włożyć nauczyliśmy się korzystać z narzędzi rozproszonej współpracy, metodologii lekkich i znaczenia kompletnej i stale rozwijanej dokumentacji technicznej. Różnorodność problemów z jakimi się spotkaliśmy pozwoliła nam zastosować w praktyce wiele nowoczesnych, multiplatformowych bibliotek i zrozumieć znaczenie wykorzystywania wzorców projektowych.
Coś się jednak nie udało...
FPTeacher nie posiada tzw. content'u, czyli zawartości. To, co zdążyliśmy przygotować na termin oddania projektu jest zaledwie demonstracją niektórych mechanizmów. Z powodu przeszacowania projektu i zbyt ambitnej koncepcji, pomimo pracy ciągłej przez ponad dwa miesiące, nie zdołaliśmy przebić się przez trzeci etap rozwoju gry - wersję grywalną. Zabrakło modułu logiki gry zawierającego również AI, kompletu asset'ów graficznych i dźwiękowych oraz etapu balansowania zawartości.
Już wyciągnęliśmy z tego wnioski.

To nie koniec rozwoju FPTEngine. Wersja 0.6 zachęca do kontynuacji, choć może nie tak dynamicznej jak dotychczas.

SPECJALNE PODZIĘKOWANIA DLA ANDRZEJA ŻURAWSKIEGO ZA PRZYGOTOWANIE I RENDERING WIZUALIZACJI AULI.

Koniec.