Programowanie zdarzeniowe – projekt

Cele projektu:

1. Zapoznanie słuchaczy z metodami:
   – tworzenia interaktywnych aplikacji z graficznym interfejsem użytkownika,
   – obsługi zdarzeń GUI generowanych przez użytkownika i zdarzeń programistycznych,
   – oprogramowania aplikacji współbieżnych i rozproszonych w metodyce zdarzeniowej,
   – oprogramowania komunikacji sieciowej w architekturze klient-serwer,
   – oprogramowania animacji,
   – wykorzystania standardowych klas Javy.
2. Implementacja przykładowej aplikacji.

Gra Dyna Blaster, np. http://www.gryjupi.pl/Gry-Na-2-Osoby/4604/Dyna-Blaster-dyna-mit-Gry-Na-2-Osoby.html

Celem gry jest niszczenie obiektów poruszających się na planszy i odnajdowanie przejść do kolejnych poziomów.

Plansza gry przedstawia labirynt z przeszkodami, po którym porusza się bohater gry (Bomber) oraz próbujące go zniszczyć potwory. Bomber oczyszcza przejścia za pomocą bomb. Wybuchająca bomba może zniszczyć obiekty w pewnym otoczeniu – potwory, przeszkody, a także samego Bombera. Niektóre obiekty, np. ściany labiryntu, są niezniszczalne, zatrzymują również falę uderzeniową. Zadaniem Bombera jest odnalezienie, początkowo ukrytego, np. pod jedną z przeszkód, przejścia do kolejnego poziomu.

Gra składa się z poziomów – plansz, przedstawiających różne układy labiryntu, przeszkód i potworów. Przejście na kolejny poziom następuje po pomyślnym ukończeniu aktualnego. Gracz ma określoną liczbę „żyć".

Gra Lunar Lander, np. https://pl.y8.com/games/lunar_lander

Celem gry jest wylądowanie statkiem na powierzchni księżyca jakiejś planety.

Plansza gry przedstawia nieregularną powierzchnię z płaskim fragmentem przeznaczonym na lądowisko. Zadaniem gracza jest doprowadzenie do pomyślnego wylądowania statku na lądowisku. Statek opada swobodnie pod wpływem grawitacji. Gracz może sterować silnikami rakietowymi spowalniającymi opadanie lub przesuwającymi statek na boki. Ruchy statku odbywają się zgodnie z prawami fizyki. Lądowanie jest pomyślne, gdy nastąpiło na lądowisku z ograniczoną prędkością (zarówno poziomą jak i pionową).

Gra składa się z poziomów – plansz, przedstawiających różne ukształtowania terenu. Kolejne poziomy różnią się szerokością lądowiska, stałą grawitacji, dopuszczalnymi prędkościami lądowania, itp. Przejście na kolejny poziom następuje po pomyślnym wylądowaniu. Rozbicie statku powoduje powtórzenie poziomu. Gracz ma określoną liczbę statków.

Ogólne uwagi dotyczące wszystkich projektów:

1. Gra ma szczegółowe zasady punktacji.
2. Program przechowuje listę najlepszych wyników.
3. Definicje wyglądu plansz, poziomów, scenariusz gry, warunki ukończenia poziomu i gry przechowywane są w plikach konfiguracyjnych; te parametry nie mogą być obliczane w programie. klasa properties
4. Funkcjonalność sieciowa obejmuje serwer przechowujący i udostępniający informacje konfiguracyjne: listę najlepszych wyników, definicje wyglądu plansz, poziomów gry, scenariusza gry oraz wszelkie pozostałe parametry aplikacji. Serwer obsługuje dowolną liczbę klientów. Serwer sieciowy jest niezależną, bezobsługową, parametryzowaną aplikacją. Aplikacja w wersji sieciowej działa także bez serwera – z wykorzystaniem lokalnych plików konfiguracyjnych. Adres serwera i numer portu są parametrami aplikacji klienta.
5. Wszelkie animacje, wynikające z logiki gry są *gładkie*, bez *migotania* i bez *przeskoków*.

6. Rysowanie obiektów graficznych i tła gry odbywa się na dwa sposoby: poprzez kolorowe kształty lub poprzez użycie plików graficznych stanowiących reprezentację graficzną obiektów gry. Na ocenę nie wpływa wybrana technika reprezentacji graficznej obiektów.
7. W trakcie gry można zmieniać wielkość okna. Pole gry dostosowuje się (skaluje) do aktualnej wielkości okna. Tryb pełnoekranowy jest niedopuszczalny.
8. Nie wolno stosować aktywnego oczekiwania na zdarzenia (*busy waiting*).
9. Program oferuje funkcję pauzy – zatrzymanie gry na dowolny czas z pozostawieniem aktualnego stanu planszy na ekranie.
10. Do programu dołączony jest skrypt uruchamiający grę (polecenie interpretera).
11. Oprogramowanie przechowywane jest w systemie kontroli wersji typu GIT, na serwerze https://gitlab-stud.elka.pw.edu.pl/. Każdy zespół zakłada w nim swój projekt i w nim przechowuje wszystkie pliki projektowe, dokumentacje, instrukcje itd. Zespół dołącza prowadzącego jego projekt do członków swojego zespołu projektowego w systemie w roli *maintainer*. Repozytorium w systemie GitLab musi nazywać się wg schematu: PROZE20L_nazwisko1_nazwisko2.

Ocenianie:

Oceniany element	Maksymalna liczba punktów
Wersja podstawowa gry	40
Elementy dodatkowe	7
Funkcjonalność sieciowa	15
Dokumentacja i instrukcja użytkownika	8

Maksymalna liczba punktów za projekt: 70. Zaliczenie projektu na podstawie:

- Działającej gry (**wraz z kodami źródłowymi wszystkich klas**).
- Dokumentacji oprogramowania:
  - Wygenerowanej programem JavaDoc (**wersja elektroniczna**).
  - Protokołu komunikacji sieciowej z komentarzem – tylko dla programów z funkcjonalnością sieciową (**wersja elektroniczna**).
- Instrukcji użytkownika wytworzonych aplikacji (**wydruk**).

Etapy zaliczenia projektu:

Każdy z elementów do zaprojektowania i wykonania należy zaliczyć w odpowiednich terminach:

1. Załóż repozytorium i umieść w nim wszelkie pliki opisujące projekt i jego założenia. Zaprojektuj szczegółowe zasady gry, narysuj szkice interfejsu graficznego. Opisz funkcjonalność aplikacji. Określ szczegółowe zasady punktacji. Zdefiniuj własne elementy dodatkowe, do **23 marca 2020 r.**
2. Zaprojektuj i zaimplementuj moduł odczytu i parsowania plików konfiguracyjnych (w szczególności pliku ze scenariuszem gry i definicją poziomu). Opracuj makietę aplikacji, tj. program prezentujący obiekty graficzne związane z logiką gry (w szczególności przykładowy poziom gry odczytany z pliku konfiguracyjnego) i interfejs graficzny użytkownika. Przedstaw uproszczony diagram klas aplikacji, do **6 kwietnia 2020 r.**
3. Do makiety z etapu drugiego dodaj obsługę zdarzeń i animację, do **27 kwietnia 2020 r.**
4. Zaprojektuj protokół sieciowy komunikacji z serwerem aplikacji – etap obowiązuje wszystkich, także tych, którzy nie zdecydowali się na realizację wersji sieciowej, implementacja zaprojektowanego protokołu nie jest wymagana, do **11 maja 2020 r.**
5. **Ostateczne zaliczenie projektu, w wyznaczonym terminie, nie później niż 19 czerwca 2020 r.**

Każdy pokazywany kod zawiera odpowiednią dokumentację JavaDoc. Brak dokumentacji JavaDoc jest równoważny niezaliczeniu etapu. Wszystkie pokazywane pliki **muszą** pochodzić z repozytorium GIT na serwerze https://gitlab-stud.elka.pw.edu.pl/. Niezaliczenie etapów 1-4 to **utrata 5 punktów** za każdy etap.