****

**Politechnika Świętokrzyska**

**Fizyka w animacji i grafice komputerowej (projekt)**

Bartłomiej Bugara, Szymon Kołodziejczyk

4ID13A

**„Sand Tetris”**

**Klon gry Tetris z fizyką**

**Sprawozdanie z prac nad projektem**

****

1. **Prace zaplanowane na pierwszy kamień milowy:**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Przygotowanie harmonogramu i podziału prac | **WYKONANO** | Plik z harmonogramem zamieszczono w serwisie Moodle. |
| 2 | Zapoznanie z podstawowym działaniem gry Tetris | **WYKONANO** | Sprawdzony został domyślny rozmiar planszy zwanej tetrionem 20x10, ilość klocków – Tetrimino, każdy z nich składa się z 4 bloków. Zwrócono również uwagę na wyświetlanie kolejnego klocka. |
| 3 | Omówienie pomysłów na projekt. | **WYKONANO** | Burza mózgów pozwoliła na wymyślenie fizycznej wersji gry tetris a także ubogacenie jej o tryb klasyczny oraz tryb wielu klocków. |
| 4 | Wybór wersji silnika graficznego – Unity. | **WYKONANO** | Wybrano najnowszą wersję LTS – 2022.3.10f1 |
| 5 | Instalacja środowiska „Visual Studio 2022” | **WYKONANO** | Przy instalacji doinstalowano pakiet „Opracowywanie gier za pomocą aparatu Unity” |
| 6 | Utworzenie repozytorium w serwisie „GitHub” | **WYKONANO** | Wykonano przy użyciu aplikacji GitHub Desktop. |
| 7 | Implementacja podstawowej logiki Tetrisa:   1. Plansza 2. Spadające bloki – klocki 3. Kolizja z podłożem | **WYKONANO** | Prototyp gry powstał 11 października. Kod zawierał definicje bloków, funkcje generowania planszy – grid’a oraz spadanie pojedynczych klocków i poruszanie nimi. |
| 8 | Implementacja fizyki piachu do klocków | **WYKONANO** | Zadanie to zostało wykonane wraz z podpunktem c) z zadania 7. Spadające bloki kolidując z podłożem (dolna krawędź gry lub inne leżące już bloki) rozsypują się w pojedyncze ziarna piasku. |
| 9 | Logika łączenia cząsteczek piachu | **WYKONANO** | Przy pomocy kilku funkcji sprawdzany jest typ bloku, jego rodzaj (kolor piachu) i ich położenie. Jeżeli ziarna rozsypane są od lewej do prawej to są usuwane. |
| 10 | Wdrożenie funkcji liczących czas i punkty | **WYKONANO** | StatsController odpowiada za zliczanie czasu oraz punktów i wyświetlanie ich. Znajdują się z nim funkcje od zatrzymywania czasu, restartu, dodawania punktów. Punkty naliczane są za każdą usuniętą linię. Początkowo była to wartość 200 punktów, aktualnie 1 punkt za każde ziarenko piasku. |
| 11 | Refaktoring kodu | **WYKONANO** | Poprawiono nazwy funkcji oraz zmiennych zgodnie z konwencją Unity. Dodano komentarze w formacie Doxygen, aby w przyszłości w prosty sposób wykonać odpowiednią dokumentację. |
| 12 | Testy i poprawa błędów | **WYKONANO** | Po przeprowadzeniu kilku testów zauważono spadki FPS, dlatego przeprowadzono poprawki mające na celu poprawić jakość gry. Usunięto zbędne listy, naddatek obliczeń w funkcjach Update(), a także zastąpiono Bubble Sort sortowaniem przy pomocy operatora OrderBy technologii LINQ. |
| 13 | Sprawozdanie z postępów prac nad projektem | **WYKONANO** |  |

1. **Podsumowanie pierwszego kamienia milowego:**

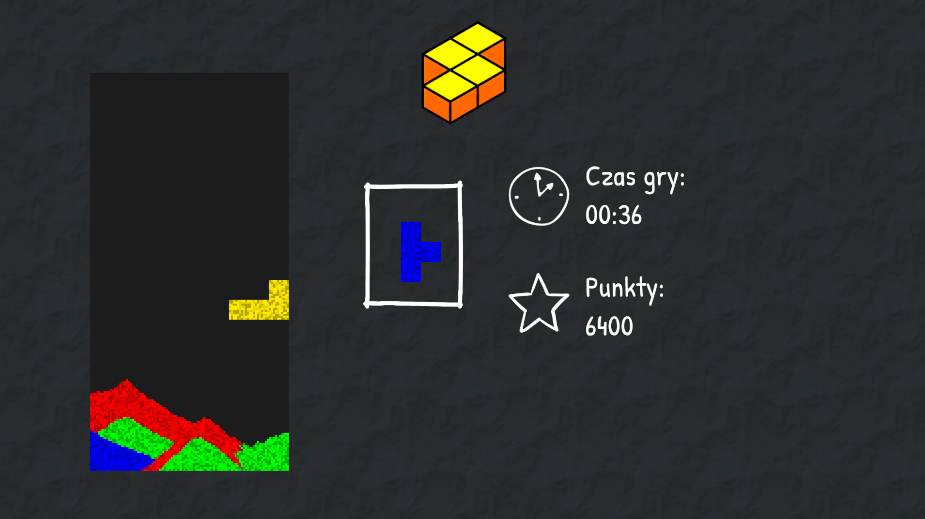
Pierwszy kamień miliowy przebiegł w pełni pomyślnie.

Wykonaliśmy wszystkie zaplanowane prace, po drodze uzupełniając je kilkoma zadaniami z kamienia drugiego, aby przyśpieszyć czas programowania projektu. Wykonana w tym okresie gra była w pełni funkcjonalna, tryb piasku działał w pełni poprawnie, został zoptymalizowany do stopnia wysokiej jakości i wydajności. Menu pozwala na rozpoczęcie gry i wybór trybu (inne zostaną dodane w drugim kamieniu) a także na ustawienie głośności dźwięków i muzyki. W grze poruszamy się przy pomocy klasycznej kombinacji WASD, klawisze AD pozwalają na ruchy klockami w lewo i prawo, klawisz S przyśpiesza opadanie zaś klawisz W powoduje rotację klocka o 90 stopni. Gra kończy się gdy na ekranie nie zmieści się więcej klocków, tj. gdy dotkniemy górnej krawędzi. Wtedy podliczany jest czas i punkty, system porównuje je z wcześniejszymi wynikami i decyduje czy umieścić je w liście najlepszej dziesiątki. Kolorystycznie utrzymaliśmy się w motywie ciemni, wraz w reflektującymi klockami w kolorach palety RGB oraz żółci. Każdy komponent gry posiada swój skrypt, w którym opisaliśmy zmienne oraz funkcje z ich parametrami. Zastosowaliśmy komentarze w stylu Doxygen, aby przyśpieszyć proces wykonywania dokumentacji.

Wykorzystane oprogramowanie:

Grafiki wykonano w programach:

* + - * GIMP – program do tworzenia grafiki rastrowej
      * Inkscape – program do tworzenia grafiki wektorowej
    - GitHub Desktop – program do obsługi systemu kontroli wersji Git



1. **Prace zaplanowane na drugi kamień milowy:**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Implementacja funkcji pauzy oraz restartu | **WYKONANO** | Klawisz ESC zatrzymuje grę, zaś w przypadku przegranej wyskakuje ekran pokazujący osiągnięty czas i wynik, z jego poziomu możemy zrestartować grę lub wrócić do menu |
| 2 | Opracowanie UI Menu oraz ekranu gry | **WYKONANO** | Wykonano grafiki przycisków, tła oraz wybrano czcionkę pasującą do gry. Ekran gry wyświetla czas, punktację oraz widok następnego klocka. |
| 3 | Implementacja klocków o innych zachowaniach fizycznych | **WYKONANO** |  |
| 4 | Implementacja efektów audio wizualnych. | **WYKONANO** | Dodano dźwięki FX do ruchów klocka, przycisków, zniszczenia linii a także przegranej. Główną ścieżką muzyczną gry zostały tradycyjne Korobeiniki, zagrane na keyboardzie i lekko wzmocnione w programie Audacity. |
| 5 | Ulepszenie funkcji punktacji o system najlepszych wyników. | **WYKONANO** | W menu głównych pod przyciskiem Wyniki znajduje się 10 najlepszych wyników. Wyświetlany jest uzyskany czas, punkty oraz tryb gry, dodatkowym parametrem jest wynik, czyli punkty podzielone przez ilość sekund, według tego sortujemy wynik od najlepszego do najgorszego. |
| 6 | Refaktoring kodu. | **WYKONANO** |  |
| 7 | Dokładne testy, wyszukiwanie błędów i ich ewentualna naprawa. | **WYKONANO** |  |
| 8 | Sprawozdanie z postępów prac. | **WYKONANO** |  |

1. **Podsumowanie drugiego kamienia milowego:**

**Przed obroną projektu:**

1. Przygotowanie dokumentacji kodu źródłowego
2. Końcowe sprawozdanie
3. Testy kodu, optymalizacja i poprawa błędów
4. Ewentualne usprawnienia i ulepszenia