Rzeszów, 30.04.2018

Dokumentacja Post Mortem

Projekt: Parallax Scrolling

Gracjan Kudra

3 EF-DI L03

1. Wstęp

Parallax Scrolling dosłownie oznacza przewijanie paralaksy jest to efekt niezgodności różnych obrazów np. tej samej strony obserwowanej z różnego jej kawałka. W szczególności paralaksa odnosi się do jednoczesnego obserwowania obiektów leżących w różnych odległościach od obserwatora, a objawia się tym, że obiekty te na wielu obrazach są oddalone od siebie o różną odległość lub też nachodzą na siebie na tych obrazach w odmiennym stopniu. Efekt ten w projekcie Parallax Scrolling, został wykorzystany w celu uzyskania wielu warstw obrazków łączących się w jeden obiekt, który dzięki w/w efektowi odwzorowuje obiekt trójwymiarowy.

2. Cel projektu

Celem projektu było utworzenie strony w technologii JavaScript, która obsłuży efekt paralaksy, wykorzystując do tego myszkę i jej gesty jak również w jakimś stopniu odwzoruje efekt na roboczych obiektach (obrazkach).

3. Realizacja projektu

Początkowe stadium projektu zaczęło się od fazy poszukiwań informacji o możliwościach technologicznych, dzięki którym efekt paralaksy byłby w ogóle widoczny. Zostały wybrane do tego odpowiednie biblioteki. Pierwsza z nich to Snap SVG jest to biblioteka, która jest odpowiedzialna za przetwarzanie obrazków z rozszerzeniem SVG (ale jak udowodnił projekt nie tylko SVG, również PNG i GIF). Druga biblioteka to TweenMax, została wybrana dlatego, że jest to rozszerzenie odpowiedzialne za animacje obiektów np. poprzednio utworzonych dzięki bibliotece Snap ale również i innych obiektów. Z początku nie była to wystarczająca biblioteka dlatego zostały podjęte dalsze poszukiwania animacji warstw obiektów, jednakże po jakimś czasie powrócono do niej ponieważ mimo wszystko spełniała wszystkie kategorie użycia jej jako głównej funkcji animacyjnej. Dalej poszukiwanie informacji skupiło się na możliwościach JavaScript-u pod względem możliwości nasłuchiwania ruchów myszki. Wybrano trzy podstawowe typy „nasłuchiwaczy", które zostały wykorzystane w projekcie. Kolejnym etapem projektu była faza projektowania, która opierała się o odpowiednie zaprogramowanie całego wyglądu jak i mechaniki strony w JavaScript. Początkowo podjęto się próby odpowiedniego przetworzenia obrazków pod bibliotekę Snap by można było z nich utworzyć warstwę dostatecznie dobrze wyglądającą i komponującą się w całość. Plany zakładały również dodania jako ostatniej warstwy obrazków pliku multimedialnego, jednakże próby się nie udały o czym w dalszej części dokumentacji. Podjęto następnie kroki w celu ominięcia wymogu plików SVG przez bibliotekę Snap. Po wymyśleniu i zaimplementowaniu sposobu podjęto próbę połączenia funkcji animacyjnej wraz z EventListenerami z JavaScript-u. Gdy i na to został wymyślony sposób podjęto się najważniejszej części czyli połączenia animacji i myszki z warstwami obrazków. Również ten etap prac zakończył się pomyślnie. Kolejny etap projektu dotyczył ewentualnych poprawek ale w dużej mierze skupiono się na następnym etapie czyli testach. Testy rozpoczęły się od podetapu sprawdzania wydajności napisanego skryptu i wykorzystanych bibliotek na różnych przeglądarkach, drugim podetapem było sprawdzenie kompatybilności wstecznej czyli na starych przeglądarkach. Po testach nastąpił etap finalizowania przystąpiono do pisani dokumentacji oraz do wykonania wykresów Gantta.

4. Testy

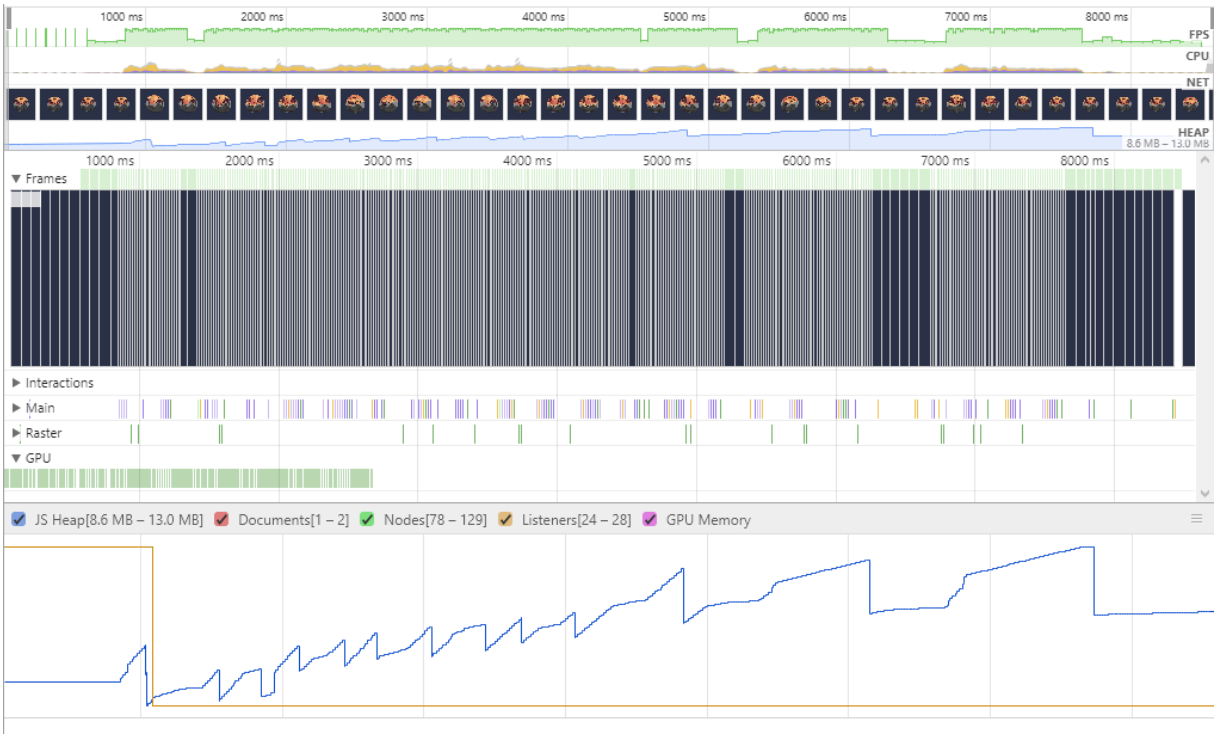
4.1 Testy wydajności skryptu

Testy wydajności skryptu opierały się o szybkość klatek na sekundę jak i skoków zużycia procesora w danym momencie czasu, testy rozpatrywane były na 3 rożnych przeglądarkach, były to:

- Opera
- Google Chrome
- Microsoft Edge.

Pierwsza jak i druga przeglądarka poradziły sobie ze skryptem bez najmniejszych zastrzeżeń, bez najmniejszego zachwiania czasem dostępu procesora bądź zbędnym generowaniem mocy obliczeniowej przez kartę graficzną, a oto ich wyniki.

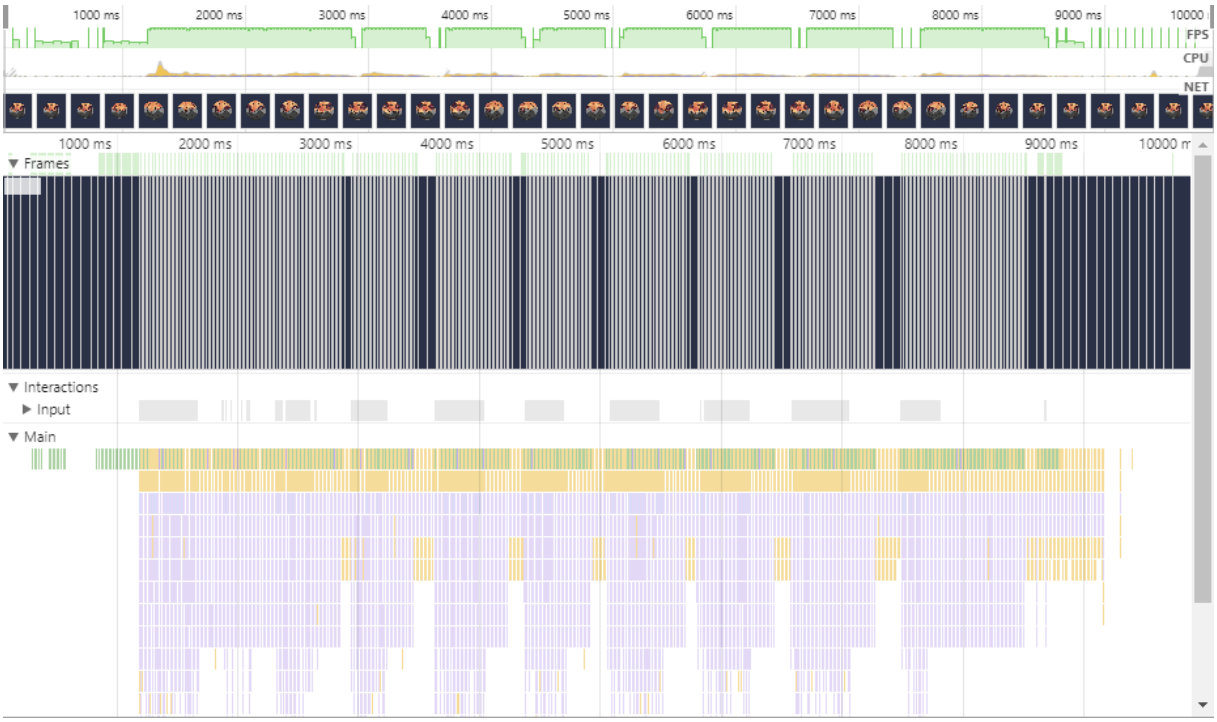
Pierwsza przeglądarka to Opera.



Rys. 1 Zrzut ekranu z Opery

Jak widać na zrzucie ekranu (Rys. 1) z testowania wydajności Opera bez najmniejszych problemów poradziła sobie z wygenerowaniem tego prostego skryptu i animacji FPS w granicach 58-60 czyli praktycznie zerowe straty. CPU obciążone mocniej tylko na początku generowania a tak prawie nie widoczne zużycie. Pamięć RAM podczas renderowania sięgnęła 13mb.

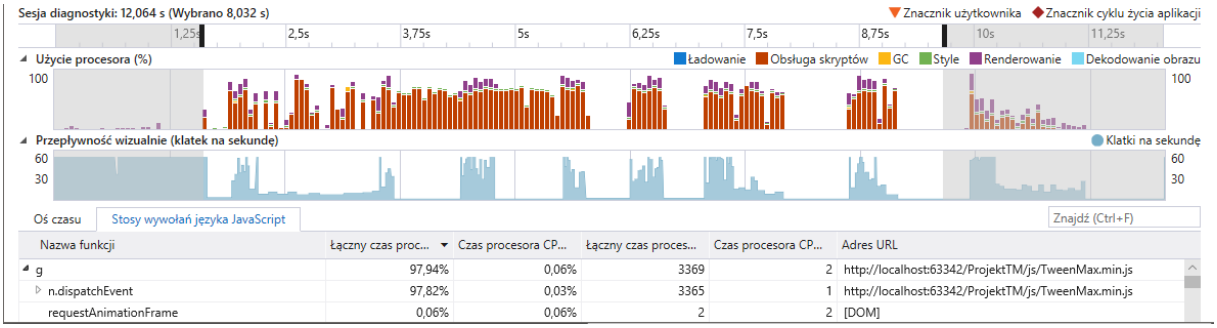
Drugą przeglądarką był Google Chrome.



Rys. 2 Zrzut ekranu z Google Chrome

Również tutaj widać (Rys. 2), że przeglądarka pod względem wydajności poradziła sobie bardzo dobrze. FPS podobnie jak u poprzednika w granicach 60, jedyne spadki jakie były to w momencie przestania ruszania się myszki. Również tutaj widoczne jest większe zużycie procesora na samym starcie poruszania się myszki natomiast dalsze ruszanie nie wywoływało żadnego większego efektu.

Ostatnią przeglądarką było Microsoft Edge czyli nowsza wersja Internet Explorera, jej wyniki jak można było się spodziewać będą niezbyt dobre ale, że aż tak złe to nie było przewidziane.



Rys. 3 Zrzut ekranu z Microsoft Edge

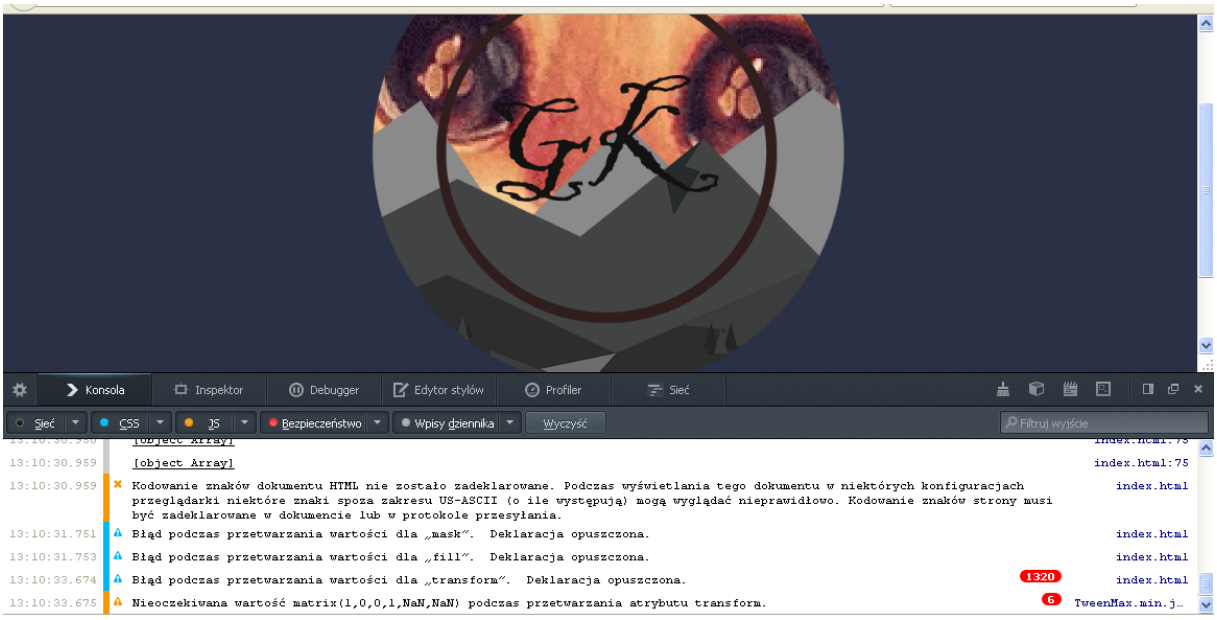
Jak widać na obrazku (Rys. 3) w momencie ruszenia myszki, przeglądarka zaczęła zużywać bardzo dużo mocy obliczeniowej, FPS-y spadły do 2 klatek i wzrastały do 60 w przypadku nie poruszania myszką w żadną ze stron. Procesor został bardzo obciążony co widać na zużyciu go przez bibliotekę TweenMax. Dla porównania we wcześniejszych przeglądarkach nie było zauważalnej różnicy w jakimkolwiek obciążeniu tą biblioteką.

4.2 Testy kompatybilności wstecznej

Testy kompatybilności wstecznej opierały się na sprawdzeniu kompatybilności skryptu na starszych przeglądarkach zainstalowanych na starszym systemie. Testowany zestaw to maszyna wirtualna z Windowsem XP i dwiema przeglądarkami :

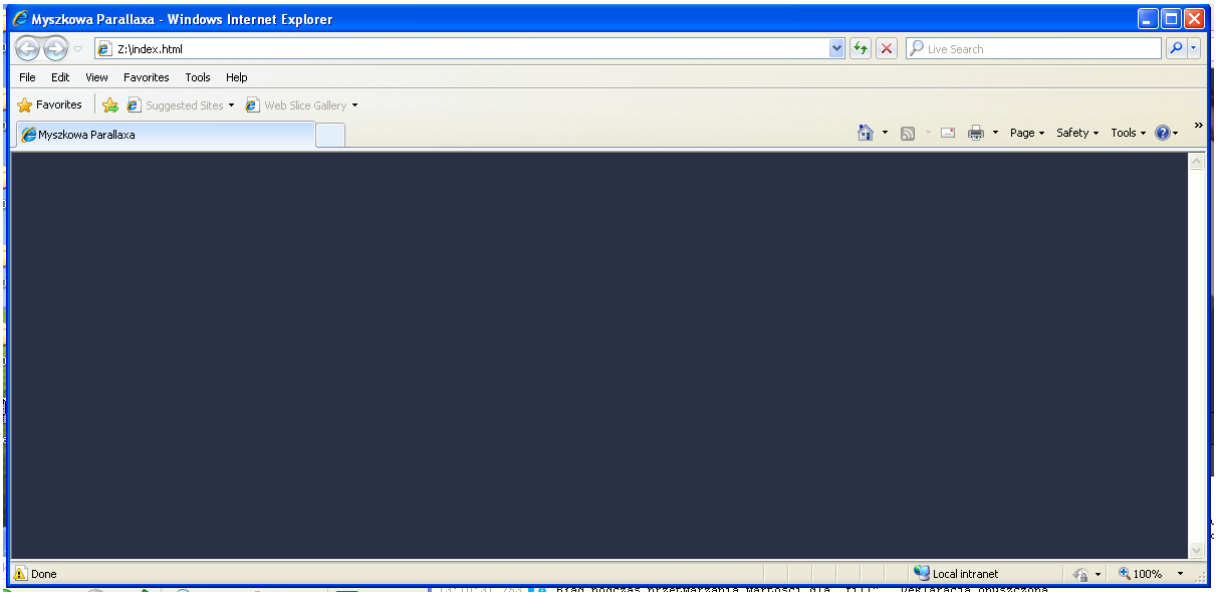
- Mozilla Firefox v 27.0.1
- Internet Explorer 8,9,10

Przeglądarka Firefox, mimo że w starszej wersji potrafiła odpalić projekt w połowie ponieważ wszelkie obrazki związane z projektem zostały przetworzone jednakże nie zostały zaanimowane w pełni, biblioteka została wczytana i animacja odpowiedzialna za najechanie i zjechanie z elementu myszką odtwarza się w pełni aczkolwiek animacja obrotu obiektu czy sam efekt paralaksy nie został w ogóle pokazany. Bardzo prawdopodobne, że pewna część biblioteki to starszy kod który przeglądarka obsługuje a ponieważ biblioteka jest ciągle aktualizowana zatem zawiera kod w nowszej odsłonie, który może być w ogóle nieinterpretowany przez przeglądarkę.



Rys. 4 Zrzut ekrany z Firefox-a

Testy w przypadku Internet Explorer-a zakończyły się stosunkowo szybko, ponieważ nie było co testować. Skrypt aplikacji stworzonej w ramach projektu nie jest zupełnie wspierany przez IE8 i starsze wersje. Dla pogłębienia testów zostały sprawdzone różne wersje przeglądarki. Okazało się że przy IE10 projekt ładuje się bez większego problemu, jednakże test dotyczył starszych wersji zatem dla starszych wersji projekt nie działa.



Rys. 5 Zrzut okienka z IE8

5. Problemy podczas tworzenia projektu

Podczas tworzenia projektu napotkano wiele błędów jednakże szczególnie para wyróżniała się najbardziej. Mianowicie były to :

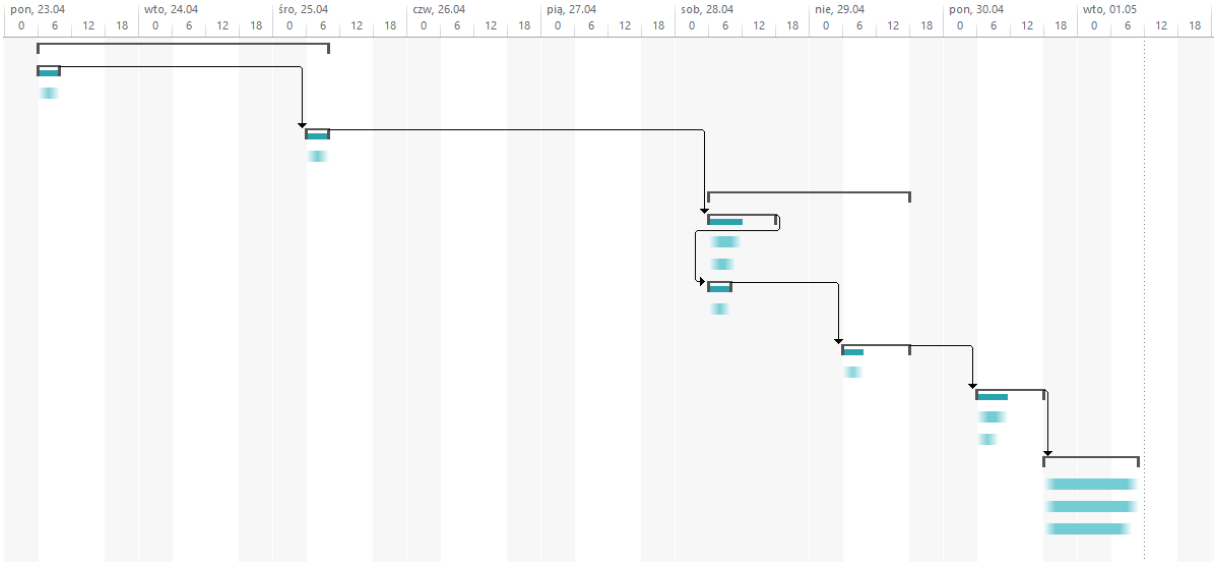
- Przetwarzanie obrazów z formatem SVG
- Zaimplementowanie obiektu multimedialnego jako ostatniej warstwy

Zaczynając od pierwszego problemu polegał on na tym, że podczas fazy przygotowywania już konkretnych obrazków pod warstwy na efekt paralaksy została napotkana nieumiejętność stworzenia i odpowiedniego wyeksportowania do formatu SVG danego obrazka. Zostało to ominięte poprzez zastosowanie innego formatu niż SVG w przypadku tego projektu były to formaty PNG i GIF. Biblioteka mimo zastrzeżeń że powinny być używane tylko pliki SVG przepuszcza przez siebie również inne formaty, format SVG byłby lepszym rozwiązaniem jednakże wykonanie tego było zbyt trudne.

Drugim napotkanym problemem, który wynikł jeszcze przed pierwszym było zaimplementowanie obiektu multimedialnego jako ostatniej warstwy została rozważana inna biblioteka. Ze względu na poziom trudności w implementacji tej biblioteki i jakiekolwiek filmu jako ostatniej warstwy, przekroczyłoby to czas trwania projektu dwukrotnie. Dlatego wybrana została biblioteka tylko do przetwarzania małych a pewnych obrazków.

6. Wykresy Gantta

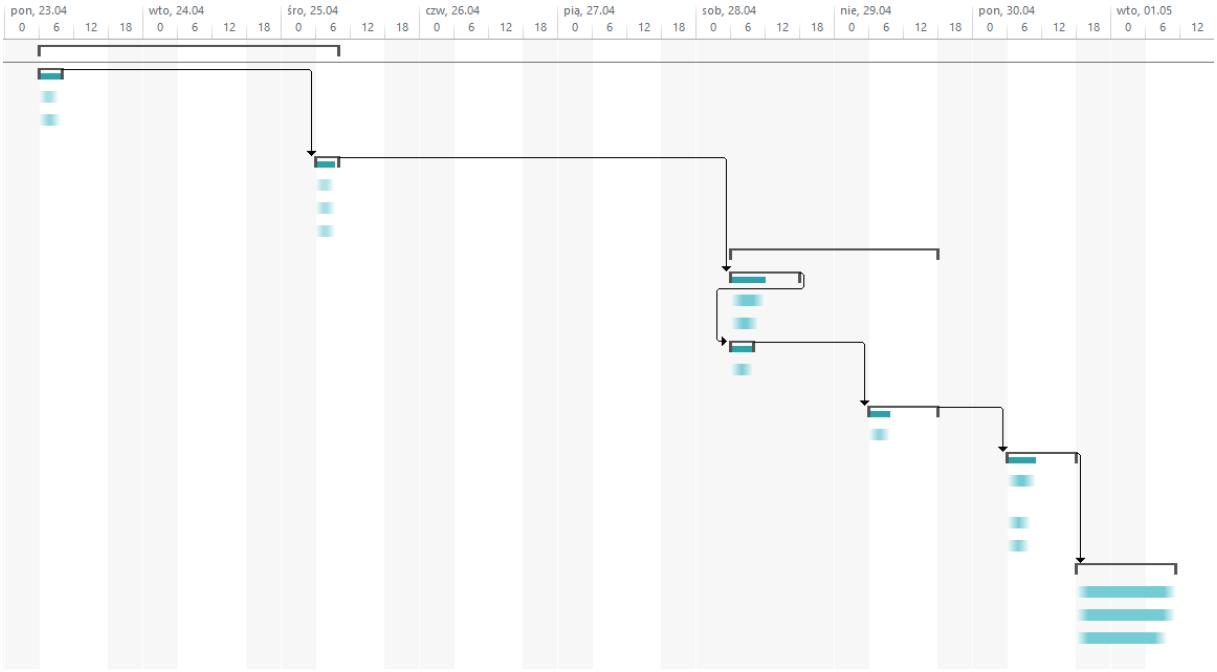
Wykres Gantta planowany kontra rzeczywisty oczywiście wykazuje różnice. Zakładany przez planowany czas na projekt to koło 15,5 godziny, jednakże rzeczywisty czas tworzenia wydłużył się do ponad 18 godzin pracy. Wynikało to głównie z problemów przy wyszukiwaniu informacji o środowisku czy możliwościach programowych, poprawianiu kodu jak również przy tworzeniu samej dokumentacji.



Rys. 6 Wykres Gantta planowany

Tryb zadani	Nazwa zadania	Czas trwania	Rozpoczęcie	Zakończenie
	Faza szukania informacji	**2,13 dn**	**pon, 23.04.18**	**śro, 25.04.18**
	Szukanie informacji o zadanym temacie	**50 min**	**pon, 23.04.18**	**pon, 23.04.18**
	Wyszukiwanie informacji dotyczących środowiska i dodatkowych bibliotek, funkcji	50 min		
	Szukanie informacji o zdarzeniach	**1 godz.**	**śro, 25.04.18**	**śro, 25.04.18**
	Wyszukiwanie informacji o mozliwosciach animacji warstw	60 min		
	Faza projektowania	**1 dzień**	**sob, 28.04.18**	**nie, 29.04.18**
	Kodowanie	**8 godz.**	**sob, 28.04.18**	**sob, 28.04.18**
	Przygotowywanie funkcji animacyjnej	3 godz.		
	Opracowanie sposobu poruszania się warstw	2 godz.		
	Zbieranie informacji na temat problemów	**1 godz.**	**sob, 28.04.18**	**sob, 28.04.18**
	Problemy dotyczące przetwarzania obrazów w animacji	1 godz.		
	Finalizowanie	**1 dzień**	**nie, 29.04.18**	**nie, 29.04.18**
	Poprawki dotyczące kodu	45 min		
	Faza testowania	**1 dzień**	**pon, 30.04.18**	**pon, 30.04.18**
	Testowanie optymalności i kompatybilności	150 min		
	Poprawki	50 min		
	Faza tworzenia dokumentacji	**0,25 dn**	**pon, 30.04.18**	**wto, 01.05.18**
	Dokumentacja kodu	120 min		
	Dokumentacja Post Mortem	120 min		
	Wykresy Gantta	40 min		

Rys. 7 Opis wykresu Gantta planowanego



Rys. 8 Wykres Gantta rzeczywisty

Tryb zadani	Nazwa zadania	Czas trwania	Rozpoczęcie	Zakończenie
	Faza szukania informacji	**2,13 dn**	**pon, 23.04.18**	**śro, 25.04.18**
	Szukanie informacji o zadanym temacie	**65 min**	**pon, 23.04.18**	**pon, 23.04.18**
	Wyszukiwanie informacji dotyczących środowiska	25 min		
	Wyszukiwanie i przetwarzanie obrazków pod parallaxe	40 min		
	Szukanie informacji o zdarzeniach	**1 godz.**	**śro, 25.04.18**	**śro, 25.04.18**
	Animowanie warstw obrazków	15 min		
	Uzyskiwanie efektu parallaxy za pomocą myszki	25 min		
	Wyszukiwanie innych możliwości animacji	20 min		
	Faza projektowania	**1 dzień**	**sob, 28.04.18**	**nie, 29.04.18**
	Kodowanie	**8 godz.**	**sob, 28.04.18**	**sob, 28.04.18**
	Przygotowywanie funkcji animacyjnej	3 godz.		
	Opracowanie sposobu poruszania się warstw	2 godz.		
	Zbieranie informacji na temat problemów	**1 godz.**	**sob, 28.04.18**	**sob, 28.04.18**
	Problemy dotyczące przetwarzania obrazów w animacji	1 godz.		
	Finalizowanie	**1 dzień**	**nie, 29.04.18**	**nie, 29.04.18**
	Poprawki dotyczące kodu	45 min		
	Faza testowania	**1 dzień**	**pon, 30.04.18**	**pon, 30.04.18**
	Testowanie strony na różnych przeglądarkach pod względem optymalności	120 min		
	Testowanie strony na starszych przeglądarkach	70 min		
	Poprawki	55 min		
	Faza tworzenia dokumentacji	**0,3 dn**	**pon, 30.04.18**	**wto, 01.05.18**
	Dokumentacja kodu	145 min		
	Dokumentacja Post Mortem	130 min		
	Wykresy Gantta	40 min		

Rys. 9 Opis wykresu Gantta rzeczywistego

7.Podsumowanie

Wykonanie projektu przebiegło pomyślnie, większość celu została wykonana. Niestety nie zostało wykonane włożenie obiektu multimedialnego jako ostatnią warstwę, ale nie to było głównym założeniem projektu. Głównym celem pozostało wciąż wytworzenie efektu paralaksy z przetworzonych warstw. Podsumowując gdyby na projekt było przeznaczone więcej czasu prawdopodobnie dałoby radę osiągnąć wszystkie cele główne jak i poboczne.