**1.Slide 2D Game Development with JavaScript**

**2.Slide - Game Development**

Razvoj video igara je počeo otprilike 1960.tih godina i traje i dan danas. Prvobitna ideja je bila simulacija određenih događaja kako bi se videli njihovi efekti, ali su video igre vrlo brzo dobile zabavni karakter.

Razvoj video igara u JavaScriptu je sve popularniji i sada već postoji veliki broj biblioteka I framework-a koji nam mogu mnogo olakšati posao ( o njima će biti reči nešto kasnije).

Početak razvoja svake video igre je ideja. Svaka igra je započela kao zamisao I onda je sprovedena u delo. Kako bi se što lakše organizovalo planiranje razvijen je dokument koji služi kao osnovni dokument pri razvoju video igara – Game Design Document (GDD). Game design document je običan tekstualni fajl koji bi trebalo napisati pre početka razvoja igre.

**3.Slide - Game Design Document (GDD)**

Osnovni delovi:

* Priča
* Heroj ili glavni junak, ostali učesnici
* Okruženje u kojem se dešava radnja
* Gameplay tj. način igranja i utisak koji igra ostavlja na igrača (brza, intezivna, opuštajuća, strašna, smirujuća...)
* Dizajn
* Muzika i zvučni efekti
* Korisnički interfejs

GDD naravno nije obavezan kao što ni njegova forma nije striktna i menja se spram potrebe igre, ali sadrži neke osnovne teze koje bi trebale biti razrađene i na kraju uklopljene zajedno u jednu celinu. Takođe GDD predstavlja vodič za razvoj igre kako bi se što manje odstupalo od prvobitne ideje.

**4.Slide - Game Development u Javascript-u**

* Najčešći I najlakši pristup je korišćenjem canvas elementa. Canvas element se koristi za iscrtavanje objekata uz pomoć JS-a. Potrebno je definisati canvas element u HTML-u i nakon toga ga mozemo koristiti pomocu JS-a (poziciju je najlakse stilizovati uz pomoc CSS-a).
* Sada ćemo proći neke osnovne koncepte koji su prisutni u game development-u kao i osnovne metode I karakteristike canvas elementa.

**5.Slide - Canvas coordinates**

Kako bi mogli da iscrtavamo objekte u canvas elementu moramo znati kako izgleda koordinatni sistem.

Koordinate u gornjem levom uglu su (0,0). Koordinate na X-osi se povecavaju sa leva na desno, dok se koordinate na Y-osi povecavaju od gore ka dole.

**6.Slide - Canvas fillRect(x,y,width,height)**

**7.Slide - Game Loop**

Dokle god je igra pokrenuta potrebno je da pratimo korisnikov unos, pozicije nasih objekata i da nakon toga ponovo nacrtamo objekte kako bi korisnik mogao da vidi promene.

**8.Slide - User input and player movement**

**9.Slide - Context2D.drawImage();**

**10.Slide - Animation**

Korsiti se niz slika koje se kontrolisano menjaju u zavisnosti od željenog broja frame-ova u sekundi. Ako kažemo da animacija ima 10 frame-ova u sekundi, znači da promenimo 10 slika u jednoj sekundi pri čemu se dobija pokret. Broj frameova animacije je nezavisan od broj frameova igre.

**11.Slide - Animation State**

Često ste spominje animation state koji se uglavnom kontroliše uz pomoć swich case-a. U zavisnosti od stanja(mirovanje, trčanje, skok...) objekta koji se animira, menjaju se slike koje se koriste, broj frame-ova kao i same karakteristike slike(širina, visina i slično).

Takodje jako bitna stvar je hijerarhija animacije. Veoma je važno znati da li animacija koja treba da se aktivira ima prednost I prekida postojeću ili ne. U igricama najčešće skakanje prekida animaciju trčanja dok to nije i obrnut slučaj (Super Mario). Animacija trčanja je stalno prisutna dok Mario ima horizontalnu brzinu (kreće se po X-os) ali kada Mario skoči animacija skoka počinje i prekida animaciju trčanja, dok to nije obrnut slučaj, Super Mario nikad neće početi da trči u vazduhu.

**12.Slide - JavaScript game dev frameworks**

Danas postoji veliki broj JS framework-a koji mnogo olakašavaju pravljenje igara. Većina početnih problema sa kojim se developer susreće su rešeni što znači da se osoba odmah može usmeriti odmah na razvoj gameplay-a I svojih ideja, bez da troši vreme na rešavanje 'matematickih' problema.

Trenutno najpopularniji su:

* Phaser
* GDevelop
* ImpactJS
* MelonJS
* Pixi.JS ...

**13.Slide - PhaserJS**

Jedan on trenutno najpopularijih framework-a. [PhaserJS](https://phaser.io/) ima gotova rešenja za prethodno spomenute probleme kao i za mnoge druga sa kojima bi se developer na početku razvoja igre susreo. Takođe imaju odličnu 'step by step' [dokumentaciju](https://phaser.io/learn) i dobar [community](https://phaser.io/community) support.

Takodje postoji i [boilerplate](https://github.com/photonstorm/phaser3-project-template) projekat koji sam iskoristio da bih postavio 2D platformer igricu. Uzimajući u obzir da je ovo bio moj prvi susret sa PhaserJS framework-om mogu da kažem da se veoma lako koristi zbog gore spomenutih karakteristika. Link do github platformera: <https://github.com/ObradovicZ/platformer>.

**14.Slide - Dobre strane Game Development-a**

* Odličan za vežbanje logike
* Odličan za unapređenje znanja Javascript jezika
* Zabavan način za rešavanje matematičkih problema
* Razvija i dopušta da pokažete kreativnost

**15.Slide - Hvala na paznji**

**Pitanja? 😀**