Dokumentacija implementacije projekta:

„Tank Wars“

Osnovna ideja revolucionarne igre Tank Wars je okršaj između tenka 1(Igrač 1) i tenka 2(Igrač 2). Igrači upravljaju tenkovima koristeći MQTT protokol. Prvi igrač koji uspije da pogodi drugog projektilom ispaljenim iz tenka je pobjednik. Kako bi što bolje objasnili implementaciju našeg programa podvući ćemo osnovne funkcionalnosti i probleme koje smo iskusili:

- UI
- Modeliranje tenkova i projektila
- Dobijanje podataka korištenjem MQTT protokola
- Detekcija kolizije između objekata
- Kašnjenje između ekranskog prikaza i stanja engin-a

Problem modeliranja tenkova i projektila smo realizovali putem klasa Tank i Missile respektivno. Klasa Tank kao atribute sadrži:

- Širinu (width)
- Visinu (height)
- Real-time pomaknuće tenka po x i y osi (speedX, speedY)
- x i y koordinate na osnovu kojih se vrši iscrtavanje (x,y)
- smijer prema kojem je okrenut tenk, koji predstavlja instancu Direction klase (direction)
- boja tenka (color)
- vektor ispaljenih projektila (missiles)

Interfejs ove klase izgleda ovako:

-Tank(int width, int height, Direction dir, int x = 0, int y = 0) – ovo predstavlja konstruktor klase

-void setColor(int color) – metoda za postavljanje boje tenka

-void draw() – metoda za iscrtavanje tenka na display

-void provjeraKolizije() – metoda za provjeru kolizije sa drugim objektima i zidovima. Provjera se izvršava vršeći test da li se pozicija tenka nalazi na kordinatama prepreka, koje su predefinisane. U prevodu, dosta ifologije.

-void updatePos() – metoda za obnavljanje koordinata tenka na osnovu brzine kretanja po osama.

-void updateScreen() – metoda koja vrši osvježavanje displaya, te poziv svih drugih funkcija koje su važne za updejt stanja igre. Jedna važna funkcionalnost ove funkcije je

upravljanje ispaljenim projektilom, te testiranje da li je taj projektil doživio koliziju sa drugim tenkom ili sa preprekom. U oba slučaja će biti urađene odgovarajuće akcije.

-void moveUp() – metoda za kretanje prema gore

-void moveDown() – metoda za kretanje prema dolje

-void moveLeft() – metoda za kretanje u lijevo

-void moveRight() – metoda za kretanje u desno

-void shoot() – metoda za ispaljivanje projektila

Klasa Missile od atributa posjeduje:

- koordinate (x, y)
- brzinu kretanja po osama (speedX, speedY)
- smijer u kojem leti, koji je instanca klase Direction (direction)

Njen interfejs izgleda ovako:

-Missile(Direction dir, int x, int y) – ovo predstavlja konstruktor klase

-void draw() – metoda za iscrtavanje projektila na display

-void updatePos() – metoda za obnavljanje koordinata projektila na osnovu brzine kretanja po osama

-void moveUp() – metoda za kretanje prema gore

-void moveDown() – metoda za kretanje prema dolje

-void moveLeft() – metoda za kretanje u lijevo

-void moveRight() – metoda za kretanje u desno

Razni problemi koji su iskoćili prilikom implementacije su rješeni koristeći navedene funkcije. Prvi od problema koji čemo razješiti je problem kašnjenje ekranskog prikaza u odnostu na stanje engin-a. Ovaj problem je riješen koristeći atribute x, y, speedY, speedX, i funkcije updatePos() i funckije za kretanje. Ideja rješenja problema je sljedeća:

Kako bi se ekran mogao updejtati između dva frame-a potrebno je sve promjene koje su se desile između frameova updaejtovati. To je urađeno na način da se u atributima speedX i speedY, pamti kretanje koje se desilo „između“ framova, dok atributi x i y, pamte stanje prethodnog frame-a. Kada dođe vrijeme za ponovo iscrtavanje poziva se funkcija „updatePos“. Zadatak ove funkcije je da izvrši updejt kordinata koji se desio „behind the scenes“, te ga prosljedi dijelu klase koji je odgovoran za iscrtavanje na ekran.

Sljedeći problem na koji čemo se osvrnuti je problem detekcija kolizije između objekata. Ovaj problem ima veoma jednostavno rješenje, ali koje zahtjeva veliku količinu raznih provjera. Naime osnovno rješenje je sljedeće: Ako se dva modela pronađu na istom mjestu na ekranu, mora se desiti neki događaj. Zbog ovoga, potrebna je velika količina testiranja da li se to upravo desilo u bilo kojem trenutku aplikacije. Ovaj problem je riješen kroz funkcije „provjeraKolizije“ koja testira pozicije svakog tenka, te funkcije „updateScreen“ koja vrši testiranje pozicije projektila.

Što se tiče povezivanja sa MQTT Dashom i problemom dobijanja podataka korištenjem MQTT protokola, u tu svhru su napisane sljedeće funckije koje su odgovorne za procesiranje primljenih signala sa MQTT Dasha:

-messageArrived_up_player1(MQTT::MessageData& md)

-messageArrived_down_player1(MQTT::MessageData& md)

-void messageArrived_left_player1(MQTT::MessageData& md)

-void messageArrived_right_player1(MQTT::MessageData& md)

-void messageArrived_shoot_player1(MQTT::MessageData& md)

-messageArrived_up_player2(MQTT::MessageData& md)

-messageArrived_down_player2(MQTT::MessageData& md)

-void messageArrived_left_player2(MQTT::MessageData& md)

-void messageArrived_right_player2(MQTT::MessageData& md)

-void messageArrived_shoot_player2(MQTT::MessageData& md)

-void messageArrived_pause(MQTT::MessageData& md)

Prilikom poćetka svog rada uspostavlja se konekcija sa MQTT brokerom, te se nakon toga rad prepušta već navedenim funkcijama. Sve ove funkcije obrađuju signale koje prime korištenjem MQTT protokola. Simpatičan problem na koji smo naišli prilikom impementacije je probem održanja signala, tj, kada funkcija primi signal 1, funkcija Tank klase koja odgovara tom signalu se treba ponavljati sve dok ne bude primljen signal 0. Problem je riješen na sljedeći način. Nakon što prime signal 1, svaka respektivna funkcija će postaviti Ticker, koji će izvršavati datu komandu sve dok opet ne bude primljen signal 0. Naprimjer, ako igrač1 prisitne dugme „Dole“, funkcija messageArrived_down_player1(MQTT::MessageData& md), će postaviti Ticker na funkciju članicu moveDown instance klase tenk1. Kada bude primljen signal 0, taj Ticker će biti detachovan.

Problem UI smo riješili na sljedeći način. Korisnike aplikacije prilikom pokretanja aplikacije dočeka MainMenu ekran sa kojeg imaju mogućnost pokrenuti igru, promjeniti boju tenkova i otvoriti About sekciju u kojoj se nalaze osnovni podaci o projektu. Za sve te mogućnosti kreirane su sljedeće metode koje vrše prikaz potrebnih podataka i objekata na display:

-void MainMenu()

-void ChangeColor()

-void About()

-void RenderScene()

-void PauseMenu()

-void NewGame()

-void WinningScreen(int color) – ovdje boja predstavlja igrača koji je pobijedio

Upravljanje ovim ekranima i prebacivanje sa jednog na drugi vrši metoda void MenuController(). Glavna ideja ovog kontrolera je da posmatra stanje ekrana, te klikove koji su izvršeni na njega. Nakon što klik bude registrovan korditante klika će biti upoređene sa kordinatama odgovarajućih opcija. Opcija čiji opseg budu sadržavao kordinate klika će biti pozvana.

Harun Čehajć 18704

Ermin Jamaković 18698

Aldin Kulaglić 18845