Projektkurs inom data- och nätverksteknik

Felix Havlind

Ylva Stråat

Yasir Riyadh

David Johansson

Projektkurs inom data- och nätverksteknik

HI1038

VT22

Sammanfattning

Är tekniskt avancerat och häftig grafik alltid bättre i spelutveckling? Svaret på denna fråga är nog nej. I sammanhang när spelare med olika livserfarenheter ska mötas i ett multiplayerspel kan det istället vara en fördel med en enkel och stilren speldesign som lättare tilltalar många olika slags kunder. Det finns också en fördel att ta lärdom av redan existerande och välbeprövade fysiska sällskapsspel som har underhållit sällskap och familjer i generationer i design av ett nytt digitalt sällskapsspel. I ambition att utveckla ett nätverksbaserat multiplayer spel med mycket bred kundbas skapades därför ett Pong spel för två eller fyra spelare med enkel och stilren retrodesign, med tanke att detta skulle attrahera kunder i åldersspann som i nuläget inte interagerar digitalt med varandra. Resultatet blev ett enkelt men lättanvänt spel redo att spelas av många.

Sammanfattning **2**

Inledning **4**
Metod 4

Produktbeskrivning **5**

Systemarkitektur **6**
Klient 7
Server 7
int playerNr 8
int arenaNr 8
int racketXPos[] 8
int ballxPos[], int ballyPos[] 9
int playerScore[] 9
int activeBalls 9
bool playSound[] 9
bool running 9
Abstrakta datatyper 10
Boll 10
Racket 11
Kollision 11
Arena 11
Audio 12
Start Menu 12
Score 12

Specifikation **13**

Slutsats **14**

Referenser **16**

Inledning

De senaste 20 åren har utvecklingen av spel och spelgrafik skett med raketfart. Dagens spel blir mer och mer realistiska och avancerade, lösningar som ungdomar för 20 år sedan knappt kunde drömma om är numera realitet. Den som är ute efter en nästintill realistisk upplevelse kan nu få det. De spel som idag erbjuds är något helt annat än 80- och 90-talets enkla streckgubbespel. Men är avancerat alltid bättre? Eller blir vi så förblindade av möjligheten att med ny teknik kunna skapa något nytt som aldrig gjorts förut att vi glömmer bort att analysera vår målgrupp och deras behov vid design av ett nytt spel. Och att vi helt missar att lära oss av det förflutna.

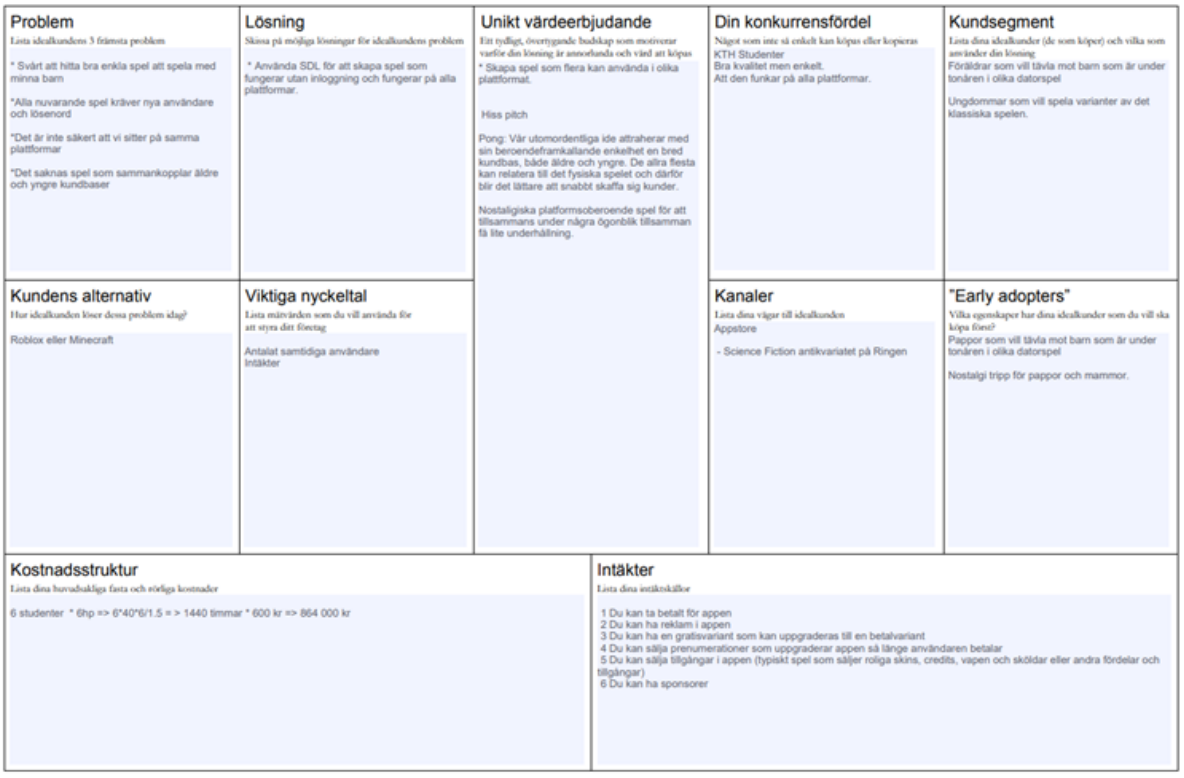
Sällskapsspel har funnits mycket länge, säkert lika lång tid som människan har funnits. Det äldsta sällskapsspelet som hittats är ungefär 4500 år gammalt och var ett spel som påminde mycket om backgammon [2] .De mest populära sällskapsspelen är ganska ofta de med få, enkla regler som gör det lätt för alla att spela, som att reglerna och spelet i sig skapar dynamik och variation som gör att det ändå krävs en hel del övning och färdighet för att bli riktigt bra på det. Man kan jämföra det med modern konst. Du kan absolut sälja en tavla med tre streck och en prick på för flera miljoner kronor, men det måste vara exakt rätt dragna tre streck och placerad prick i rätta nyanser för att fånga ögat och suga in dig i tavlan. Tavlan suger in många olika sorters människor och alla får sin egen upplevelse av den. Det skadar inte heller att konstnären är välkänd, som till exempel Mirò i ovan nämnda exempel. Ett enkelt och avkopplande digitalt sällskapsspel som attraherar alla ålderskategorier borde därför designas med samma princip. Enkla regler som ändå skapar någon form av komplexitet, simpel men stilren och rogivande design.

Metod

En av de grundläggande syftena i projektet var användandet av metoden agil utveckling. Denna projektmetodik går ut på att utveckla större delar av spelet samtidigt för varje etapp för att enkelt kunna reflektera kring delarna [1]. Skillnaden från vattenfallsmetoden där man enbart fokuserar på en del i taget. Gruppen har regelbundet varje vecka samlats antingen på plats eller via Discord för att ha etappmöten för att reflektera kring projektets gång samt problem. Under dessa möten har produktloggen skapats, uppdaterats och olika användarberättelser har diskuterats om hur de ska konstrueras.
https://gits-15.sys.kth.se/havlind/ideal-engine/projects/2

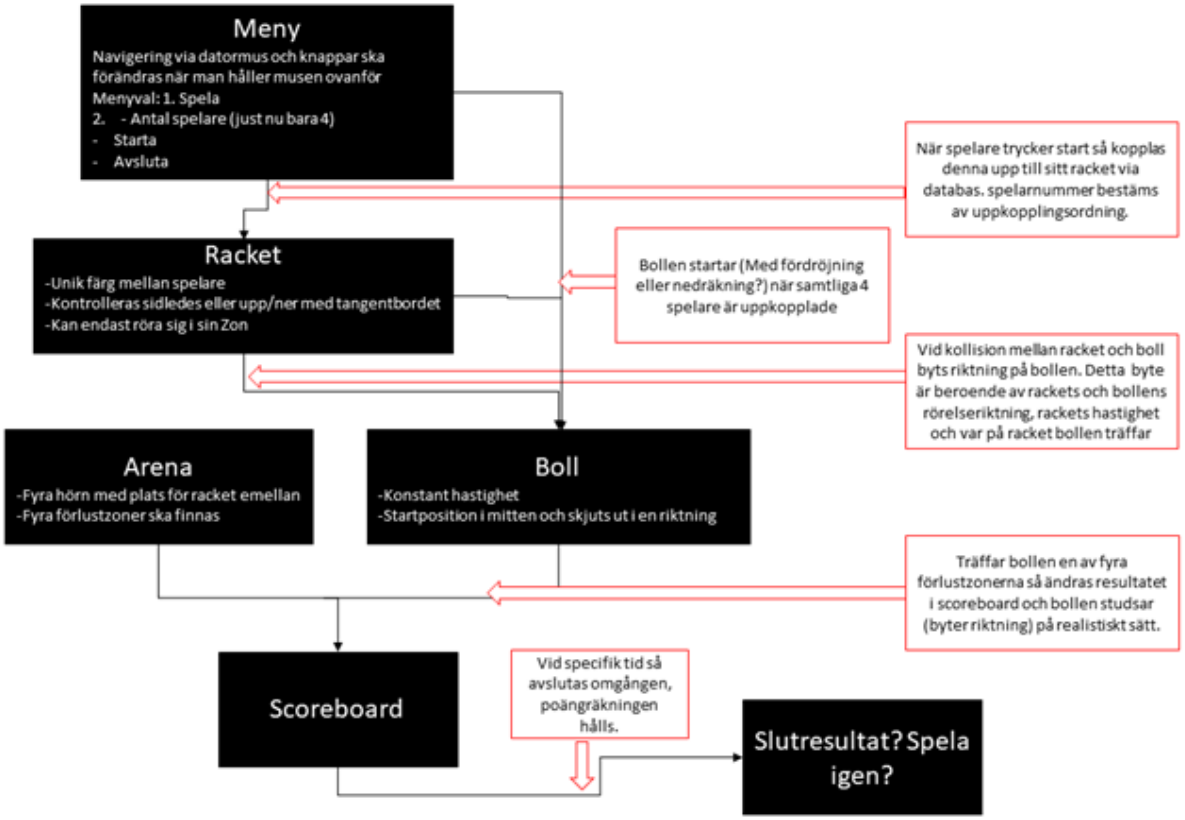
Produktbeskrivning

När spelidén utkristalliserades så ansågs det vara en fördel att använda ett redan beprövat och populärt koncept. Genom att digitalisera ett redan existerande och uppskattat fysiskt spel skulle de flesta presumtiva kunder redan känna till hur det digitala spelet skulle spelas och även i de flesta fall ha en positiv bild över hur själva spelupplevelsen skulle kunna tänkas vara. Allt detta bara genom att bli exponerade av själva namnet på spelet, vilket skulle vara en stor marknadsmässig fördel. Att använda en redan välbeprövad spelidé skulle också ge stora fördelar i analys över vilken kundkrets spelet skulle kunna tänkas intressera. I detta fall eftersöktes ett spel som skulle kunna spelas och uppskattas både av barn och vuxna. Efter noga övervägande valdes Pong för två till fyra personer då det ansågs vara lämpligt för många olika sorters sällskap, även familjer innehållandes stort åldersspann. För att ytterligare bredda kundkretsen så valdes att koda i SDL, vilket möjliggjorde att spela spelet tillsammans från olika plattformar. Detta är en funktion som ofta saknas i mycket spelutveckling idag. Att göra ett användarvänligt och enkelt nätverksbaserat multiplayer spel ansågs också medföra stora fördelar då det öppnar upp helt nya kundkretsar, till exempel våra covidisolerade äldre skulle kunna spela med sina barnbarn och på så sätt hålla en närmare och mer naturlig digital kontakt än bara med telefonsamtal.



Systemarkitektur

Arkitekturen skulle vara enkel men tilltalande, ungefär liknande den tavla av konstnären Mirò som beskrevs i inledningen. Rätt mängd och rätt placerade streck och prickar i genomtänkta färger för att symbolisera spelets olika objekt. Dessa olika objekt skulle vara programmerade så att förändring i ett objekt inte nämnvärt skulle påverkade resten av delarnas funktion. Spelet skulle göras med denna komponentbaserad programmering eftersom det skulle ge större möjlighet till förändring och vidareutveckling av spelets funktioner och design, samtidigt som att buggar inte skulle fortplanta sig i lika stor utsträckning i övriga delar av spelet. Boll skulle starta i mitten och fångas upp och studsa vidare mellan spelarnas racket och på Arenan. Missad boll skulle via fyra förlustzoner ge minuspoäng och varje spel skulle hålla på under viss angiven tid. Menyn för val av spel skulle vara enkel men tydlig och poängtavla bredvid respektive racket skulle finnas för poängräkning. För att få en tydlig bild av detta så ritades ett beroendediagram upp (se nedan).



Ett av projektets krav var att spelet måste ha kapacitet för en sorts nätverksmodell som möjliggör det att spela på ett nätverk mellan flera datorer. Detta innebar i praktiken att man behövde programmera en egen server som kan kommunicera mellan klienter på ett nätverk.

Klient

Klienten innefattar själva spelet och är där all data som servern skickar renderas och representeras visuellt. Datan som klienten skickar till servern är minimal och innefattar enbart rackets position i structen “dataStructRecieve. Variabeln uppdateras varje gång klienten ger en input inmatning och flyttar på racketet vilket sedan paketeras in i en “dataStrucktRecieve” och skickas till servern.

```
typedef struct
{
    int racketXPos;
} dataStructRecieve;
```

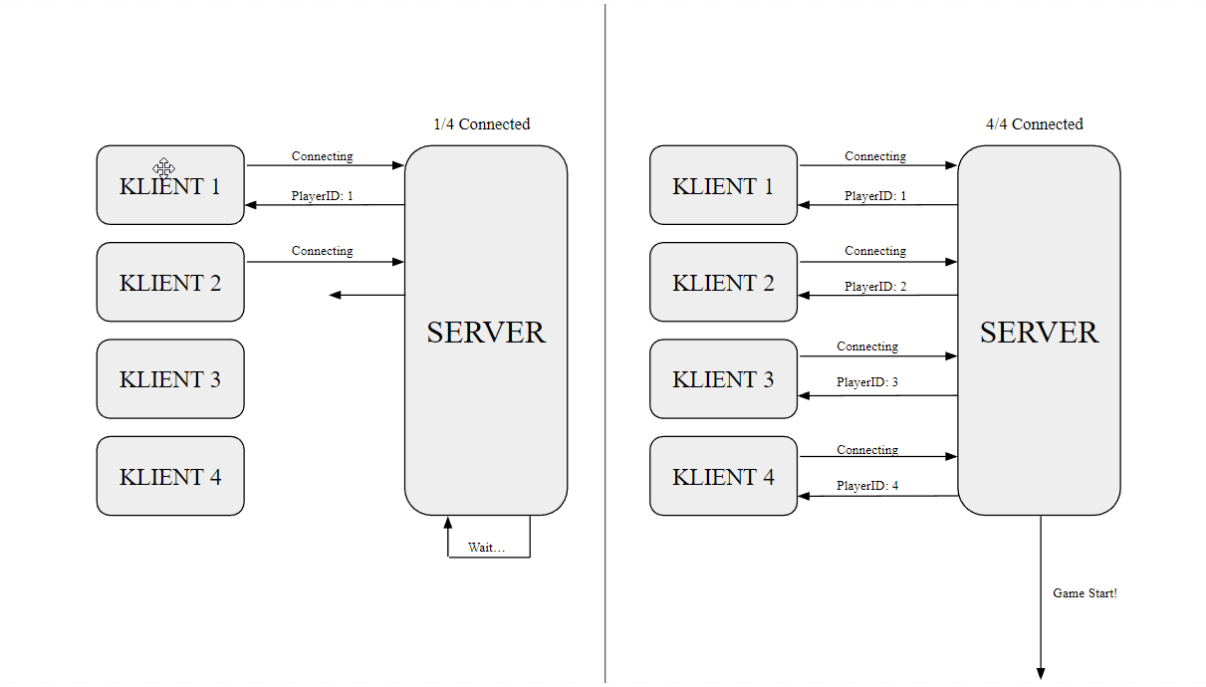
Server

Servern som utvecklats till projektet är baserad på SDL_net.h biblioteket samt använder sig av UDP-protokollet för att skicka data mellan klienterna. Detta valet gjorde vi eftersom spelet är ett realtidsspel som kräver att klienterna får ny data från servern i princip hela tiden för att spelet ska hållas uppdaterat mellan alla aktiva klienterna. Kommunikationen kan delas upp i två delar, den första delen där den initiala uppkopplingen mellan server och klient. Sedan den andra delen där själva spelet är igång och servern uppdaterar alla klienterna med nödvändig data.

Uppkopplingprocessen sker enligt UDP-protokollet där klienten initierar processen genom att startas. Klienten skickar ett irrelevant datapaket som enbart fungerar som en signal att den vill koppla upp sig. Servern svarar på detta paket genom att skicka tillbaka ett paket med innehållet “dataStructConnect” som är en struct med innehållet “int playerNr” samt “arenaNr”. Dessa variabler definierar två viktiga komponenter för att spelet ska fungera

```
typedef struct
{
    int playerNr;
    int arenaNr;
} dataStructConnection;
```

korrekt. Servern väntar tills alla fyra klienter har kopplat upp sig och endast då startas spelet.



int playerNr

Denna variabel användes både av server och klient och helt enkelt definierar klienten relativt till andra klienter.

int arenaNr

Denna variabel bestäms när servern startas och innan någon klient kan koppla upp sig och definierar vilken arena som bestämts att spela på.

Den andra delen av servern är då spelet är igång och servern skickar data till alla klienterna kontinuerligt. Servern skickar paketet “dataStructSend” till alla aktiva klienter som innehåller all data som krävs för att spelet ska fungera.

```
typedef struct
{
    int racketXPos[MAX_CLIENT];
    int ballxPos[MAX_BALL];
    int ballyPos[MAX_BALL];
    int playerScore[MAX_CLIENT];
    int activeBalls;
    bool playSound[MAX_SOUND];
    bool running;
} dataStructSend;
```

int racketXPos[]

Innan servern skickar datapaketet till klienterna, tar den emot ett paket från

klienterna. Detta paket innehåller positionen på varje respektive klients racket i x-led. Efter severn tagit emot positionerna läggs de i arrayen “racketXPos[]” som sedan skickas till varje klient.

int ballxPos[], int ballyPos[]

Innehåller boll positionen i x, samt y-led. Detta är även en array som gör det möjligt för servern att ha flera bollar samtidigt på spelplanen.

int playerScore[]

Servern håller koll på varje spelares insläppta bollar med hjälp av denna array som skickas med för att uppdatera varje klient om hur matchen går.

int activeBalls

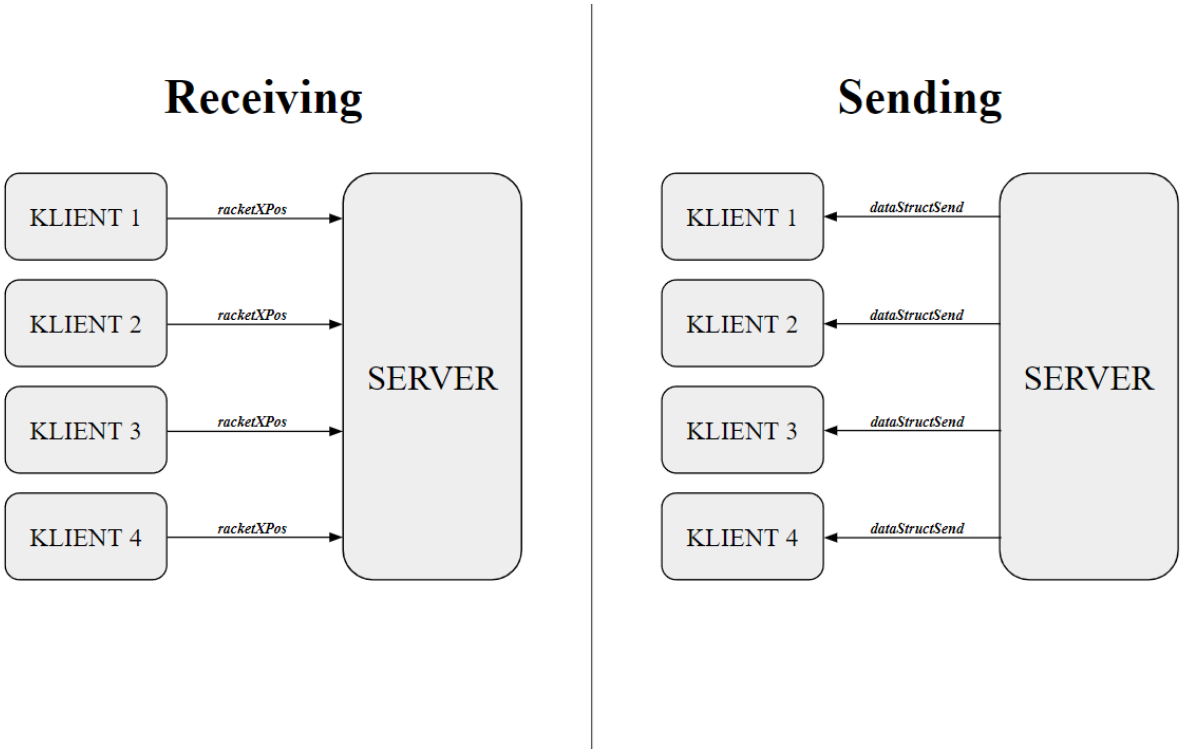
En simpel integer som representerar antalet aktiva bollar som severn har skapat. Denna variabel är viktig för klienterna att veta så tillräckligt många bollar renderas.

bool playSound[]

Ljudsystemet är även delvis strukturerat på serversidan. Eftersom servern hanterar kollisioner så skickas även ett sant eller falskt värde med för varje ljudeffekt. Exempelvis när en boll träffar ett racket så slås en av “playSound[]” värdena till “True” som klienterna sedan översätter till att spela en viss ljudeffekt. Efteråt slår klienterna, samt severn värdet tillbaka till “False” för att hindra att flera av samma ljudeffekt spelar flera gånger i rad.

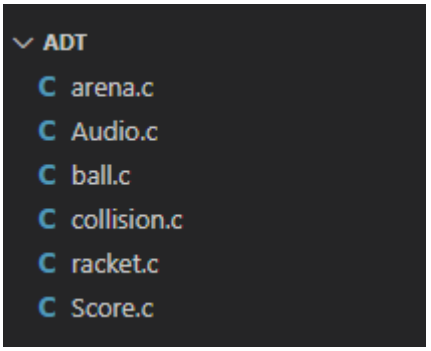
bool running

En enkel sant eller falskt variabel som severn sätter baserat på ifall spelet fortfarande är igång (True) eller om matchen är över (False).



Abstrakta datatyper

De abstrakta datatyperna är de olika objekten som klienten och servern använder sig av. Själva filerna fungerar som samlingspunkter för allting som är relevant för just det objektet och kan endast manipuleras med hjälp av funktioner. Filen brukar struktureras med en definitions del, vilket innehåller alla relevanta variabler som ett objekt har. Men även funktioner av olika slag som används för att manipulera dessa variabler för olika anledningar.



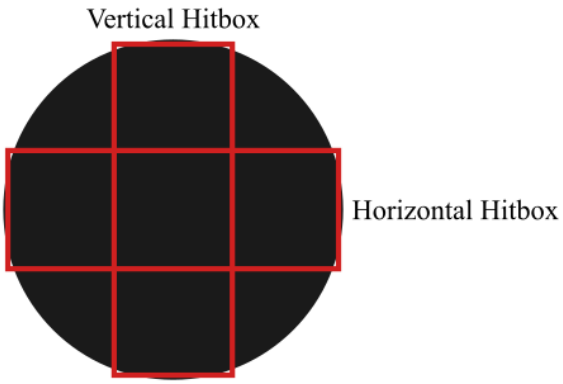
Boll

Boll ADT:n innehåller definitioner, och funktioner som bollen i spelet använder sig av. Definitionerna är bollens position i x och y led, storlek, hastighet, samt två rektanglar. Rektanglarna används för att ge bollen en fysisk representation så att den kan samarbeta med våran kollisions ADT. Funktionerna som bollen använder sig av är operationer att sätta och få definitions värdena, skapa bollen, samt ta bort bollen. Denna ADT används enbart på servern medan på klient sidan så representeras den enbart med en rektangel med en textur.

Racket

Racket innehåller liknande funktioner till vad bollen har. Dens definitioner innefattar x-position, y-position, höjd, bredd, textur, och en rektangel. Funktioner som manipulerar racket är, skapande av racket, sedan ett flertal få och sätt funktioner för de olika definitions variablerna.

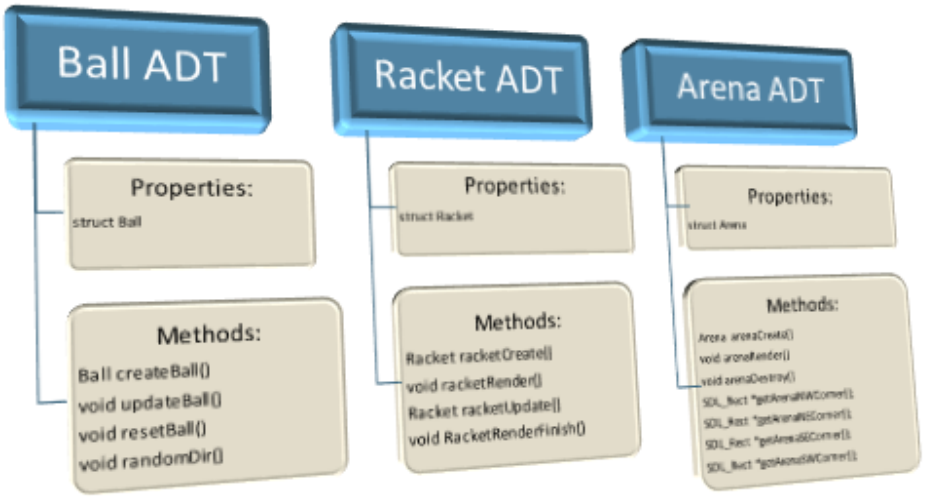
Kollision



Kollisionen mellan bollen, racket, och andra objekt på spelplanen är något som servern kollar kontinuerligt. Genom att delvis utnyttja SDL funktionen "*SDL_HasIntersection*" som returnerar '1' ifall två rektanglar kolliderar, och att representera bollens fysiska kropp med två rektanglar så blev resultatet ett fungerande och enkelt sätt att kontrollera kollisionen. Den ena rektangeln kontrollerar bollens kollision på den vertikala axeln och när den kolliderar med en annan rektangel så inverteras bollens hastighet i y-led. Medan den horisontella rektangeln fungerar likadant fas på x-axeln.

Arena

Arena ATD:n är definierad av flera rektanglar. Positionen av dessa rektanglar baseras på den tidigare nämnda "arenaNr" variabeln som positionerar rektanglarna i tre olika sorters mönster som definierar de olika arenorna. Även inkluderar denna ADT en funktion som "får tag" på minnesadressen på dessa rektanglarna. Denna funktion används exempelvis för att se till att racketet inte går för långt åt höger eller vänster.



Audio

Audio ADT innefattar hela ljudsystemet som spelet använder sig av. Den är baserad på SDL biblioteket *SDL_mixer.h* och innefattar funktioner som initialiserar ljudsystemet, och hanterandet av ljudeffekter. Musiken som spelas i menyn är också baserad från denna ADT och hanteras med samma funktioner som ljudeffekterna. ADT:n innefattar även en funktion för att städa upp och släppa fri som minne som allokerades från initialiseringen.

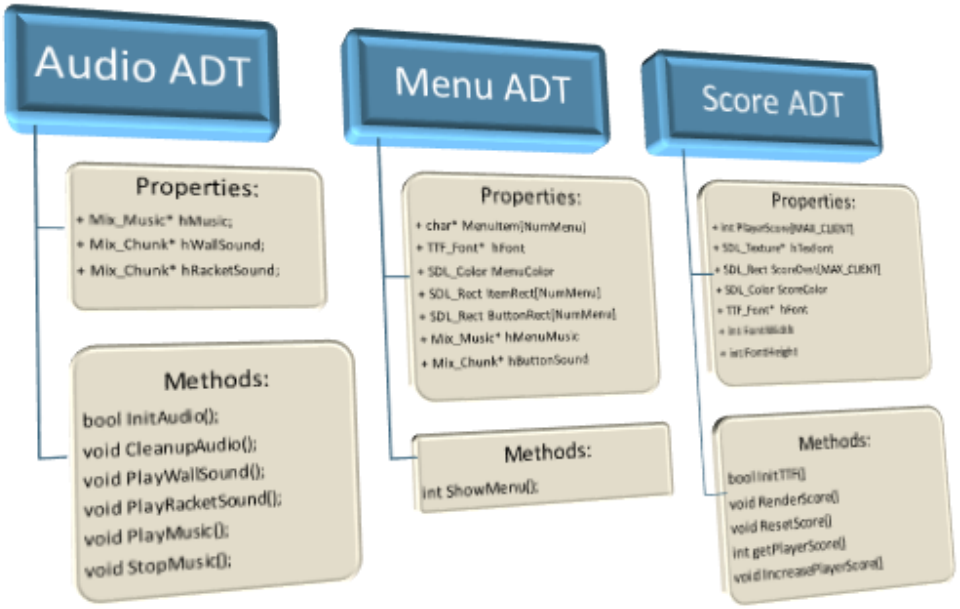
Start Menu

Denna ADT används av servern och visar en renderad meny som startar innan själva uppkopplingen och erbjuder användaren ett par olika alternativ. Här kan användaren ändra på arenaNr, antal spelare, avsluta, och även starta servern. Detta möjliggörs genom interagerande mellan musen och olika knappar på menyn. Menyn har bara en sorts funktion vilket är att visa meny.

Score

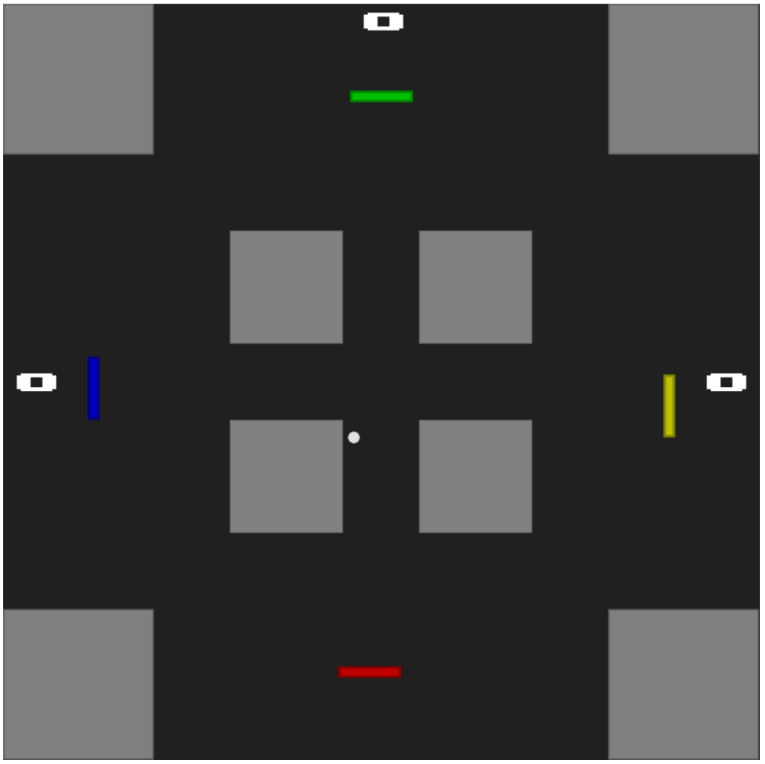
Under en spelmatch så räknas varje insläpp boll för att hålla koll på vem som vinner. Arrayen “playerScore” skickas från servern till klienterna i varje paket och innehåller de totala

insläppta bollarna för varje klient. Klienterna använder denna array för att rendera dessa nummer på spelplanen för att visualisera hur matchen går, detta görs med hjälp av Score.c ADT:n. Utöver renderingen så innefattar denna ADT även en initialiseringsfunktion som placerar ut rektanglar på rätt plats på spelplanen för "playerScore" för respektive spelare.

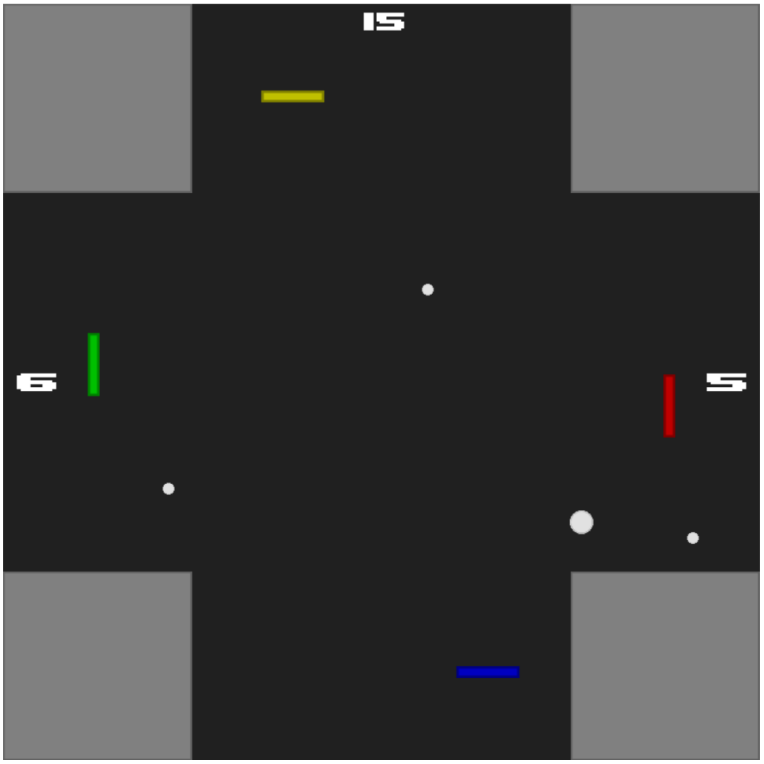


Specifikation

Det slutgiltiga projektet blev ett spel som stödjer 4 spelare med olika arenor att välja mellan. Detta görs med hjälp av en spelmeny som startar innan servern med knappar som går att interagera med musen. Spelet spelas via nätverk enligt en server/klient-modell som är programmerad i C kod med SDL_net biblioteket. Även spelet är programmerat i C kod och använder sig av SDL bibliotek så



som image, net, mixer, ttf. Med hjälp av en server sidad poängräknare som håller koll på spelets gång.



Slutsats

Kursen har tyvärr kantats av stora problem inom gruppen. En stor del av gruppens medlemmar både anslöt och hoppade av sent, och en medlem drabbades av långvarig sjukdom. Dessa problem resulterade i mycket tappad tid, kunskap och programmerad kod. Vi hade kontinuerlig dialog via Discord och digitala möten varje vecka, men vi kunde tillsammans anammat det Agila arbetssättet [1] lite bättre och genomfört sprintarna med fler fysiska och digitala möten. Det hade resulterat i bättre planering och tydligare prioritering och användning av användarberättelser. Vi hade också kunnat fånga upp de problem med programmeringen som uppkom tidigare om vi varit mer synkade, och det hade resulterat i bättre stöttning och mindre förlust av viktig tid och kod. Kanske det hade underlättat om vi tillsammans i början försökt med extreme programmering och programmerat på en dator tillsammans. Det hade varit till stor hjälp om vi lyckats få till möten med vår handledare, då vi verkligen hade behövt stöttning i de tidigare nämnda utmaningarna. Det hade också minskat den förvirring som fanns om vad som förväntats av en i kursen och när (hur) saker och ting ska vara gjorda, något som också har kostat på i tid och energi. Dessa problem har

resulterat i att några av de initiala idéerna som till exempel möjlighet att få spela mot AI har fått stryka på foten. Stämningen och tonläget i gruppen har dock varit ganska trevlig och positiv, någonting som möjliggjort en konstruktiv dialog och på så sätt har arbetet ändå kunnat drivas framåt. Överlag har det varit en relevant upplevelse som mycket väl har illustrerat vilka problem man skulle kunna möta på en arbetsplats och har gett oss stor erfarenhet hur dessa kan bemötas. Resultatet i form av innovativ produkt lämnar kanske lite mer att önska, men som det nämndes i början av kursen så var det inte heller riktigt målet med denna kurs utan det var att vi skulle tränas i att arbeta i ett Agilt projekt. Retrospektivt så kanske det hade varit bättre att titta lite närmare på några av de mer fantasifulla idéerna om spel som vi hade initialt, men vi hade en idé om vilka presumtiva kunder vi ville nå och då passade Pong bättre in i affärsmodellen.

Referenser

[1] A. Cajander. Agil Utveckling HE1041 VT22-1 Mikrodatorteknik, projektkurs (instructure.com) Nerladdad 1 April 2022

[2] K. Pietroszek, Z. Agraraharja and C. Eckhardt, "The Royal Game of Ur: Virtual Reality Prototype of the Board Game Played in Ancient Mesopotamia," 2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW), 2021, pp. 647-648, doi: 10.1109/VRW52623.2021.00206.