2022-12-06:

Idag bestämde jag mig för att starta ett projekt.

Jag ska programmera ett spel som jag kallar Dodge

Shooter. Spelet ska gå ut på att avancera sin

utrustning för att kunna bekämpa större fiender.

Spelaren kommer att befinna sig i en värld (2d), där

den kan besöka en affär för att köpa nya prylar, t.ex.

vapen, skyddsutrustning osv. Prylarna kommer att kosta

spelpengar, som man kan få genom att bekämpa fiender.

Svårare fiender kommer att vara mer lönsamma att bekämpa.

Den största utmaningen kommer vara den grafiska. Jag kommer att använda mig av bilder, som jag ritat i förslagsvis Paint, för att bygga upp omgivningen. I programmet kommer koordinater för ”fysiska” objekt finnas lagrade i förslagsvis en array. På så sätt kan spelaren interagera med statiska saker i bilden, t.ex. kollidera.

Spelarens framsteg kommer automatiskt att lagras lokalt på datorn.

En eventuell vidareutveckling kan vara att lägga till en story, konversationer osv.

Inledningsvis tänker jag börja med fundamentala funktioner i

spelet, så som att röra på gubben.

2022-12-08:

Gubben kan nu röra på sig åt alla håll med hjälp av tangenter på tangentbordet.

Just nu kollar inte programmet efter kollisioner, men jag fixar det när jag lägger till saker i omgivningen i framtiden.

Så här ser funktionen som uppdaterar gubben position ut:

function updatePosition() {

let streif = 1; // Declares the variable streif and assigns it the value 1

   // If the player is about to move diagonally, set streif to the square root of 0.5

   if ((upPressed && (leftPressed || rightPressed)) ||

(downPressed && (leftPressed || rightPressed)))

{

      streif = Math.sqrt(0.5);

   }

   if (upPressed) player.y -= player.speed \* streif; // if the up-arrow is pressed, update the player y pos

   else if (downPressed) player.y += player.speed \* streif; // if the down-arrow is pressed, update the player y pos

   if (rightPressed) player.x += player.speed \* streif; // if the right-arrow is pressed, update the player y pos

   else if (leftPressed) player.x -= player.speed \* streif; // if the left-arrow is pressed, update the player y pos

}

Jag tror dock att detta bara kommer att användas när spelaren möter en fiende. När spelaren befinner sig i ”världen” så vill jag att världen ska flytta sig runt spelaren istället.

Jag skapar en booleon som håller koll på ifall det är strid eller inte:

let fight = false;

2022-12-13

Igår gjorde jag en testbild i GIMP, 3000 x 3000 pixlar. Bilden ska vara bakgrundsbild och miljö för spelet, och det funkar ganska bra. Spelaren kan röra sig över bilden. Bilden skapades på detta sätt:

let backImg = new Image(); // Creates new image object, named backImg

backImg.src = "img/TestMap2.png"; // Assigns a image source

backImg.xPos = midWidth - 1500; // Gives it a new propeties, xPos and yPos

backImg.yPos = midHeight - 1500;

backImg.collidables = []; // Gives it another property, an array with all collidable objects

För att spelaren ska kunna kollidera med objekt i bilden, så kommer objekt att lagras i en array, som sedan kommer att loopas igenom för att upptäcka kollision. Det ska finnas två typer av objekt:

{t: ”r”, x: , y: , w: , h: } och {t: ”c”, x: , y: , r: }. Den första beskriver en rektangel, och den andra en cirkel. Programmet kan detektera vilken typ det är, genom att kolla på t.

Nu har jag plottat ut alla hörn och kanter på ett hus i bilen. Jag ska nu lägga in dessa i arrayen:

// Gives backImg another property, an array with all collidable objects

backImg.collidables = [

   // Shop house

   { t: "r", x: 1065, y: 2685, w: 360, h: 30 }, // Top left wall

   { t: "r", x: 1065, y: 2715, w: 30, h: 240 }, // Left wall

   { t: "r", x: 1065, y: 2955, w: 870, h: 30 }, // Bottom wall

   { t: "r", x: 1905, y: 2715, w: 30, h: 240 }, // Right wall

   { t: "r", x: 1575, y: 2685, w: 360, h: 30 }, // Top right wall

];

Nu behöver programmet kolla ifall det kommer att ske en kollision, innan spelarens position updateras. För att göra detta kan jag baka in det i funktionen updatePos(), men då skulle det bli en väldigt stor och stökig funktion. Istället tänker jag skapa en funktion checkCollision() som sedan sätter booleons till true om spelaren får röra sig åt det hållet. Detta har nackdelen att jag behöva en till uppsättning booleons som jag skulle kunnat klara mig utan, men jag tror det blir snyggare.

Här finns det möjlighet för mig att gå tillbaka och skriva om i framtiden ifall jag inte gillar det.

Här är de två uppsättningar booleons jag pratar om:

let upPressed = false;

let downPressed = false;

let rightPressed = false;

let leftPressed = false;

let spacePressed = false;

let moveUp = false;

let moveDown = false;

let moveRight = false;

let moveLeft = false;

Nu ser huvudloopen ut såhär:

function main() {

   checkCollision();

   updatePosition();

   drawFrame();

   requestAnimationFrame(main);

}

Jag har lyckats någorlunda med kollision. Men jag måste skriva om/färdigt en annan dag. Väldigt stora if-satser. Känns som att man borde kunna göra detta på ett bättre sätt…

Nu tänker Mauritz ”Avståndsformeln!” men jag jobbar inte med kollision av cirklar, dvs det finns inte bestämda avstånd (radier)…

Pust.. Nu funkar det med kollison och jag är nöjd med hur koden ser ut. Jag har skapat ytterligare en funktion, checkOverlapp() som kollar ifall det kommer att ske ett överlapp mellan spelaren och ett objekt i spelarens nya position. Om funktionen returnerar True så görs inte förflyttningen. Detta kollas i två steg, först efter förflyttning i x-led och därefter i y-led. Detta är viktigt för att undivka buggar där man kan gå igenom saker diagonalt, och för att man fortfarande ska kunna gå åt ett visst håll, även om en av riktningarna är blockerade.

function checkOverlapp(ax, ay, aw, ah, bx, by, bw, bh) {

   return ax + aw >= bx && ax - aw <= bx + bw && ay + ah >= by && ay - ah <= by + bh;

}

Funktionen funkar bara om spelarens koordinater beskriver spelarens mittpunkt, vilket en cirkel automatiskt gör med ctx.arc() i canvasen.

Jag kan använda samma funktion för att starta uppdrag genom att spelaren ställer sig på ett visst ställe eller andra interaktioner. Det behöver inte handla om kollision.

Jag har en del att kommentera efter idag. Jag får göra det vid nästa tillfälle.

**2022-12-14**

Jag tänker börja med att skriva en funktion som placerar spelaren på en viss relativ position till backgrundsbilden (dvs flyttar bilden så att spelaren hamnar där). Funktionen kommer att användas för att placera spelaren på olika platser och för att möjliggöra att spelaren kan befinna sig precis intill en vägg. Just nu kan spelaren ibland inte gå närme väggen än 5px ( spelarens hastighet ), för då stoppar kollisionen spelaren. Funktionen kommer alltså användas för att placera spelaren precis intill alla väggar (kollisions objekt).

Här är funktionen:

function moveToRelPos(x, y) {

   backImg.xPos = midWidth - x;

   backImg.yPos = midHeight - y;

   player.relPos.x = x;

   player.relPos.y = y;

}

Nu tänker jag rita en gubbe och använda den bilden som spelare.

Här är gubben jag ritade lite snabbt:



Eftersom att spelaren nu inte längre är en cirkel, så inser jag att jag kommer att fixa om lite i kollisionen, men förhoppningsvis är det inte några stora problem.

Jag kopierade filerna och skrev om lite för att det ska fungera för en spelare som inte är rund. Nu funkar det också, men det blev lite mer svårtläst. Ska försöka fixa till det lite snyggare nu.

Jag misstänker att nästa steg kan vara ganska jobbigt att programera, men programmet är hitils väldigt flexibelt, så jag tror jag har gett mig själv goda förutsättningar.

Nästa steg är nämligen att gubben ska vända på sig beroende på vilket håll han rör sig i. Jag ska vända på bilden, byta plats på spelarens höjd och längd ( player.h och player.w) och se till att gubben endast gör det om det är möjligt. Dvs om jag står mot en vägg och sedan vill gå i sidleds kanske gubben rör sig i sidles utan att vända på sig. Detta kan vara en mekanik som går att utnyttja i själva spelet också.

Fy och fy och fy och usch. Nu kan spelaren roteras och alla värden ändras som det ska. Det värsta var ju att för att rotera spelaren så måste man rotera hela canvasen med ctx.rotate(), som också roterar hela koordinatsystemet. För att rotera spelaren ritar jag först bakgrunden, sedan roterar jag canvasen, anpassar koordinaterna efter hur canvasen roterades så att spelaren fortfarande ritas på samma ställe, och sedan ritar jag spelaren, och roterar tillbaka canvasen.

// Draw player

   if (rotateRight) {

      // if current rotation is Right

      ctx.rotate(Math.PI / 2); // rotate canvas context by 45 degrees

      let x = player.y - player.imageH / 2;

      let y = -1 \* player.x - player.imageW / 2;

      ctx.drawImage(player.image, x, y); // draw the player

   } else if (rotateDown) {

      // if current rotation is Down

      ctx.rotate(Math.PI); // rotate canvas context by 180 degrees

      let x = -1 \* player.x - player.imageW / 2; // adjust x and y coordinates

      let y = -1 \* player.y - player.imageH / 2;

      ctx.drawImage(player.image, x, y); // draw the player

   } else if (rotateLeft) {

      // if current rotation is Left

      ctx.rotate((3 \* Math.PI) / 2); // rotate canvas context by 270 degrees

      let x = -1 \* player.y - player.imageW / 2; // adjust x and y coordinates

      let y = player.x - player.imageW / 2;

      ctx.drawImage(player.image, x, y); // draw the player

   } else {

      // if not left, right, or down. Current rotation is Up. No adjustment needed.

      ctx.drawImage(player.image, player.x - player.imageW / 2, player.y - player.imageH / 2);

   }

   if (rotateRight) ctx.rotate((-1 \* Math.PI) / 2); // rotate canvas context back

   else if (rotateDown) ctx.rotate(-1 \* Math.PI); // rotate canvas context back

   else if (rotateLeft) ctx.rotate((-1 \* (3 \* Math.PI)) / 2); // rotate canvas context back

Variablarna rotateRight, rotateDown osv styrs av en funktion kallad rotatePlayer(), som också ändrar spelarens relevanta höjd och bredd som används vid kollision.

Programmet är nu 300 rader kod långt. Det känns för mycket för vad programmet gör, men det är inte så himla mycket kvar att skriva. Att skapa en hel omgivning fylld av objekt kommer endast vara ett par rader kod, eftersom att resten redan hanteras automatiskt.

**2022-12-15**

Jag ritade en ”bakgrundsbild” som kommer att vara den textur utanför spelytan:

En bild som visar svart, vit

Automatiskt genererad beskrivning

Jag har ritat alla bilder i programmet GIMP, och börjar bli mer bekannt med programmet. Nu är varje ”pixel” åtta pixlar i höjd och bredd. Det är något jag vill ha konsekvent i spelet, bara för att det är snyggt. Dock vet jag inte vilken bredd jag vill ha på pixlarna. Det går lätt att konvertera bilderna till olika storlekar utan att förlora kvalité.

Jag vet dock inte riktigt än hur jag vill få in den som bakgrundsbild, jag tänkte jag att kunde sätta typ canvas.style.backgroundImage = ””; till bilden men det verkar inte fungera.

canvas.style.backgroundImage = "img/BackgroundStone1.png"; // funkar ej

Det är lite konstigt tycker jag eftersom att man kan sätta backgroundColor, men aja jag får klura på det sen, det är inte väsentligt.

Jag gjorde också en ljudfil i garageband på mobilen som bakgrundsmusik. Det visar sig att webbläsaren kräver att användaren på något sätt har interagerat med sidan för att en ljudfil ska kunna spelas, så jag gjorde på detta sätt:

let musicOn = false;

let backgroundMusic = new Audio("sounds/gameTrack1.m4a");

backgroundMusic.loop = true;

let musicTimer;

let playSound = () => { backgroundMusic.play(); };

document.addEventListener("keydown", () => {

   if (!musicOn) {

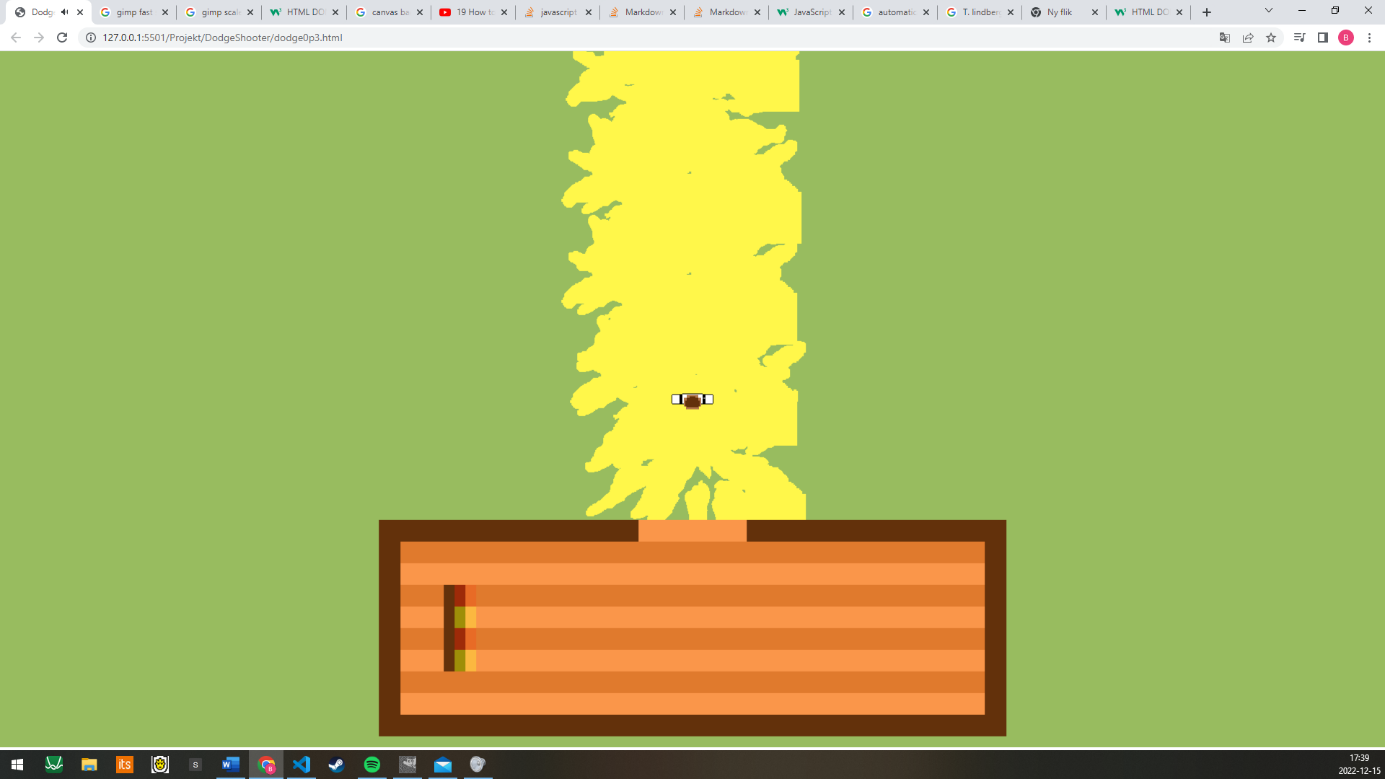
      playSound();

      musicOn = true;

   }

});

Nästa steg kan vara att börja göra stridsmekaniker, dvs kunna skjuta och sikta osv. Föresten kan jag lägga in en screenshot på hur spelet ser ut i nuläget:



Jag tänker rita upp en mer omfattande miljö nu, och lägga in alla koordinater i koden så att kollision fungerar med de nya objekten också 😊

Såhär långt kom jag med ritandet idag:En bild som visar text

Automatiskt genererad beskrivning

**2022-12-16**

Jobbar just nu med omgivningen. Ritat en grotta uppe i vänstra hörnet. Den kan användas som en ingång till en annan miljö.

En bild som visar text, elektronik, dator, visa

Automatiskt genererad beskrivning

När jag lägger in nya objekt att kollidera med funkar det SÅ bra. Det är riktigt smidigt att röra sig runt saker, och om gången är för smal för att gubben ska kunna gå igenom, så vänder sig gubben om och går i sidleds. Riktigt snyggt.

Nästa steg är att ”skärmen” endast ska följa gubben (om man får utrycka sig så) om spelaren inte befinner sig vid kanterna av världen. Borde kunna fixa det idag eller imorgon. För att det ska fungera behöver jag två sätt att röra på gubben. Just nu flyttar jag bilden istället för gubben, men uppdaterar vilken pixel på bilden som gubben befinner sig (det som jag tidigare kallat för gubbens relativa position). Det behöver också finnas ett sätt att flytta på gubben istället för bilden, så att gubben går runt på skärmen. De här två lägena kommer att byta fram och tillbaka beroende på var spelaren befinner sig. På så sätt kan jag se till att man aldrig ser kanterna av bakgrundsbilden.

Jag upplevde också lite lagg :O Men jag hade väldigt många flikar igång så jag hoppas på att det var det, men jag kan behöva se över min kod och ser var jag kan göra programmet snabbare, t.ex. behöver jag inte kolla kollision med alla objekt över hela kartan. Något jag kan försöka göra i framtiden är att kolla kollision med endast närliggande objekt.

**2022-12-17**

Idag har jag kommenterat allt i min kod, och gjort det möjligt att byta mellan två sätt att flytta spelaren på. Nu ska jag implementera den funktionen så att spelaren in kan se utanför bakgrunden såhär:



Fågan är var i koden som jag ska kolla om spelaren befinner sig nära kanterna. Jag tror jag gör en ny funktion för det som får köras av main() loopen. Jag struntar i att spelaren inte ska kunna gå utanför bilden tror jag, istället lägger jag in objekt i det andra sättet att kolla efter kollision.

**2022-12-18**

Okej. Igår lyckades jag fixa så att när man når kanterna av världen så stannar skärmen och istället rör sig gubben fritt och allt funkar bra, förutom en funktion som jag skapade förut, moveToRelPos(x,y). Den funktionen använde ju jag för att flytta gubben precis intill objekt när gubben var på väg att kollidera. På så sätt kunde gubben ställa sig precis intill väggar osv. Nu funkar inte den, eftersom att den inte ”tänker” på att backgrundsbilden inte får flyttas ur skärmens vy.

Intuitivt tänker jag att jag i moveToRelPos(x,y) kollar ifall nästa förflyttning av bakgrundsbilden kommer att ligga helt i skärmens vy eller inte. Det gör jag med isWithin(ax,ay,aw,ah,bx,by,bw,bh), och om den inte är det så får jag konstruera ett sätt som fungerar. Dvs en if-else sats som kollar resultatet av isWithin(), och agerar efter det. Det behöver nog också framgå av isWithin åt vilket håll som det blir överlapp. Annars vet inte moveToRelPos(x,y) hur den ska placera bilden.

Jag lyckas inte få till det. Jag gör om allt jag gjort och skjuter upp det. Det är inte ett stort problem, men vore snyggt att få till det.

Nu tänker jag att jag ska rita några bilder som kan få det att se ut som att gubben rör på sig, sprites.

Okej sprites var mycket enklare än vad jag trodde att det skulle vara. Kort sagt använde jag mig av en setInterval loop med en räknare. Räknaren ökade med ett efter varje loop. Med hjälp av räknaren bytte loopen source på player.image.src till nästa bild osv, som lagras i en array. Jag har 12 bilder i animationen, och de är namngivna på följande sätt: PlayerWalk1.png, PlayerWalk2.png, PlayerWalk3.png osv. Dessa läggs automatiskt in i arrayen i början av programmet med hjälp av en for-loop.

let animationCount = 0;

let animatePlayer;

let isAnimatingPlayer = false;

function animatePlayerFunc() {

   {

      player.image.src = player.walkAnimationImages[animationCount];

      if (animationCount >= 11) {

         animationCount = 0;

      } else {

         animationCount++;

      }

   }

}

function startPlayerAnimation() {

   if (!isAnimatingPlayer) {

      animatePlayer = setInterval(animatePlayerFunc, 50);

      isAnimatingPlayer = true;

   }

}

function stopPlayerAnimation() {

   if (isAnimatingPlayer) {

      clearInterval(animatePlayer);

      player.image.src = "img/PlayerIdle.png";

      isAnimatingPlayer = false;

   }

}

Funktionerna startPlayerAnimation() och stopPlayerAnimation kallas på i updatePos() om någon rörelse av spelaren sker, eller inte.

Ibland flimrar gubben, som att bilderna inte hinner läsas in, eller så är det för att timingen med framen inte matchar. I framtiden kanske jag inte använder mig av en setInterval loop.

**2022-12-19**

Jag tror nu mer på att det är inladdningen av bilderna som får gubben att flimra. Detta grundar jag på att det endast sker i början, och sedan förekommer det inte igen. Det finns säkert något sätt att ladda in bilderna i förhand innan de används, jag får googla på det senare. Just nu är jag sugen på att göra det möjligt att interagera med omgivningen ytterligare.

När spelaren trycker på tangenten ”e” så ska programmet med hjälp av funktionen checkOverlapp() kolla ifall spelaren befinner sig vid ett ställe där något går att interagera med. T.ex. en gubbe som man kan prata med osv. Jag ska lagra koordinater och andra egenskaper för dessa ytor i en ny array som kommer att tillhöra objektet backImg. På så sätt gör jag den enklare för mig när jag sedan ska byta bakgrund.

**2022-12-19**

Jag håller på med små buggar och försöker se vad i koden som orsakar dem.

Jag trodde verkligen inte att jag skulle lyckas lösa buggarna. Jag har nämligen klurat på det länge hemma också. Det var fel på lite olika sätt. Ett fel var i min main() loop. Där låg en funktion på fel ställe. Det gjorde att en funktion använde sig av information som inte hade uppdaterats än. Sedan var det lite strul med moveToRelPos() när skärmen (canvasen) låg i kanten av bakgrundbilden. Det är löst. moveToRelPos() blev lite större och jag behövde göra en ny funktion getOverlappDirection() som kollar vilken sida det är som överlappar. Den informationen användes i moveToRelPos().

Jag tänker att jag kommenterar moveToRelPos() de sista 10 min på lektionen nu.

**2022-12-23**

Idag ska jag jobba vidare med att skapa interaktioner mellan spelaren och omgivningen.

**2022-12-27**

Jag kommer inte ihåg om jag glömt att skriva i loggboken eller helt enkelt inte jobbat med programmet på några dagar. Hursomhelst.

Jag har fixat så att spelaren kan skjuta projektiler. När spelaren skjuter så skapas ett nytt objekt, som innehåller x, y, vx, och vy, alltså koordinater och hastighet i x och y komposanter. Objekten lagras i en array. Såhär går det till:

/\* This function checks if the player has

   pressed the spacebar, and if the game is

   currently in fight-mode.

   If so, the shoot() function will be called,

   also with and interval to shoot with the desired

   firerate. \*/

function checkWeaponFired() {

   if (spacePressed && fight && isFire == false) {

      shoot();

      shootInterval = setInterval(shoot, player.weapon.fireRate \* 100);

      isFire = true;

   } else if (!(spacePressed && fight)) {

      clearInterval(shootInterval);

      isFire = false;

   }

}

function shoot() {

   let shoulderX = player.x + player.w / 2; // Stores shoulders x position on the screen

   let shoulderXRel = player.relPos.x + player.w / 2; // Stores the shoulders x position over the background image

   let vx; // Declare a variable for velocity in x-direction

   let vy; // Declare a variable for velocity in y-direction

   // This if-statement checks if the mouse coordinates are numbers

   if (isNaN(mouse.x) == false && isNaN(mouse.y) == false) {

      let dist = Math.sqrt((mouse.x - shoulderX) \*\* 2 + (mouse.y - player.y) \*\* 2); // Calculate distance between mouse and player

      vx = -(shoulderX - mouse.x) / dist; // If velocity = 1, this value defines velocity in x-direction

      vy = -(player.y - mouse.y) / dist; // If velocity = 1, this value defines velocity in x-direction

   } else {

      // If the mouse coordinates doesn't have valid values, shoot up

      vx = 0; // Set velocity in x-direction to 0

      vy = -1; // Set velocity in y-direction to -1

   }

   // Create a new object and place it in the projectiles array

   projectiles.push(new CreateProjectile(shoulderXRel, player.relPos.y, vx \* player.weapon.speed, vy \* player.weapon.speed));

}

function CreateProjectile(x, y, vx, vy) {

   this.x = x;

   this.y = y;

   this.vx = vx;

   this.vy = vy;

}

Varje frame körs updateProjectilePosition() som uppdaterar positonen på varje objekt i arrayen projectiles:

function updateProjectilePosition() {

   for (let i of projectiles) {

      i.x += i.vx;

      i.y += i.vy;

   }

}

Slutligen ritas alla projectiler i drawFrame() :

...

   // draw projectiles

   for (let i of projectiles) {

      let x = currentEnv.xPos + i.x;

      let y = currentEnv.yPos + i.y;

      ctx.beginPath();

      ctx.arc(x, y, 3, 0, 2 \* Math.PI);

      ctx.fill();

   }

}

Nu ska jag med hjälp av array.splice() ta bort de projectiler som befinner sig utanför bakgrunden. Detta gör jag i funktionen updateProjectilePosition().

Detta verkar fungera, men jag va skeptisk till att ta bort objekt ur en array samtidigt som jag loopar igenom varje index. Ett problem skulle kunna resultera i att enstaka projektilers position inte uppdateras i den framen. Men det är inget jag märker av, och jag kommer lämna det så.

function updateProjectilePosition() {

   for (let i of projectiles) {

      i.x += i.vx;

      i.y += i.vy;

      if (i.x < 0 || i.x > currentEnv.width || i.y < 0 || i.y > currentEnv.height) {

         projectiles.splice(i, 1);

      }

   }

}

Jag kommer lägga tid på att kommentera all kod senare. Detta för att jag inte är säker på om jag vill behålla koden än.

Jag kom på att jag kan skjuta genom väggar, det vill jag inte. Därför lade jag till detta. Detta skriker dock LÅNGSAMT.

function updateProjectilePosition() {

   for (let i of projectiles) {

      i.x += i.vx;

      i.y += i.vy;

      if (i.x < 0 || i.x > currentEnv.width || i.y < 0 || i.y > currentEnv.height) {

         projectiles.splice(i, 1);

      } else {

         for (let j of currentEnv.collidables) {

            if (checkOverlapp(i.x - 3, i.y - 3, 6, 6, j.x, j.y, j.w, j.h)) {

               projectiles.splice(i, 1);

               break;

            }

         }

      }

   }

}

Lyckligvis blir det inte så jättemånga projektiler samtidigt.

Vad ska jag då göra näst? Jag tänker att jag börjar lägga till pengar, och kanske menyer som man kan öppna genom att besöka specifika platser på kartan. T.ex. en meny för att uppgradera sitt vapen. Jag börjar med att rita en bild på en peng i GIMP.



Så blev resultatet.

Här är en bok som man kommer kunna uppgradera sin gubbe i. Också ritad i GIMP:

En bild som visar text, gul, byggnad, rullgardin

Automatiskt genererad beskrivning

Nu tar jag en paus.

**2022-12-28**

Jag har precis väldigt enkelt gjort så att man kan öppna boken ovan om man befinner sig vid smeden. Detta görs med funktionen checkInteraction(), som sätter booleonen bookOpen till true om man befinner sig vid smeden. Om bookOpen är true, så ritas boken i varje frame i drawFrame().

Nu ska jag rita flera saker i boken. Bland annat ett kryss som man kan stänga ned boken med, och olika uppgraderingar.



**2022-12-29**

Idag har jag lagt till en armor- och healthbar. Healthbaren övergår också gradvis till rött när spelaren har låg hälsa.

En bild som visar text

Automatiskt genererad beskrivningEn bild som visar text, poängtavla

Automatiskt genererad beskrivning

Med ctx.fillStyle = ”rgba(…)” kunde jag manipulera färgerna och också göra de genomskinliga så att spelaren har en chans att se cad som befinner sig bakom.

Nu planerar jag att möjliggöra uppgraderingarna i boken, och kanske rita en ny ”bok” som kommer att kunna öppnas i huset. I den boken kanske man kan sälja saker för att tjäna pengar eller ngt sånt. Största utmaningen i framtiden kommer vara att programmera fiender.

Det kommer alltid finnas saker att lägga till, det är bara en fråga om hur mycket jag kommer hinna. Får jag ett komplett spel kan jag göra en minimap, mer avancerade konversationer och massa andra småsaker.

Nu ligger programmet på 750 rader kod.

**2022-12-30**

Idag hade jag tänkt lära mig lite om hur fiende AI vanligtvis programmeras.

Jag hittade inte något väldigt nytt.. Jag har lite olika alternativ hur fienden ska fungera.

Antingen så är det bestämt exakt vad som fienden ska göra och upprepa, t.ex. först släppa bomber, sen flytta på sig till slumpmässig plats, skjuta tre skott mot spelaren och upprepa. Eller så ska fienden hela tiden anpassa sig eftersituationen och ta beräknade beslut.

Jag tror jag börjar med den enklare varianten, och enkla fiender som helt enkelt rör sig mot spelaren osv.

Först måste jag ha en plats där man kan slåss mot fiender. Jag gör ritar nog en ingång till en annan värld, och finslipar på hur man byter karta.

Jag vill också kunna sälja och köpa saker, t.ex. mediciner osv. Spelaren kommer därför behöva ett inventory. Det ska jag göra någon gång i framtiden också.

Nu har jag ritat ett slott på bakgrundsbilden och lagt in koordinater i programmet så att spelaren kolliderar och kan byta bakgrundsbild.

En bild som visar text, elektronik, skärmbild, dator

Automatiskt genererad beskrivning

Nu behöver jag rita en ny spelyta för slottet.

**2023-01-01**

Första gången jag skriver 2023 :D

Jag har precis gjort en väldigt enkel fiende, men det funkar till att börja med. Helt enkelt så kommer fienden att jaga spelaren, om spelaren befinner sig inom en viss radie. Ifall spelaren och fienden överlappar så kommer spelaren ta skada. För att göra detta skrev jag tre nya funktioner: createEnemy(), updateEnemy(), damagePlayer(). createEnemy() används för att skapa ett nytt objekt och lägga den i en array. Funktionen har parameterar som x och y position, hastighet, skada osv. updateEnemy() kallas på varje frame i huvudloopen. Den funktionen uppdaterar fiendens position om spelaren befinner sig i närheten, och anropar damagePlayer() om det sker en krock mellan fiende och spelare. Här är funktionerna:

function createEnemy(x, y, w, h, dmg, health, speed, fireRate, range){

   currentEnv.enemies.push({ x: x, y: y, w: w, h: h, vx: 0, vy: 0, dmg: dmg, health: health, speed: speed, fireRate: fireRate, range: range });

}

function updateEnemies() {

   // for every enemy in current enviornment

   for (let i of currentEnv.enemies) {

      let diffX = i.x - player.relPos.x; // calculate distance between player and enemy over  in x-direction

      let diffY = i.y - player.relPos.y; // calculate distance between player and enemy over  in y-direction

      let dist = Math.sqrt(diffX \*\* 2 + diffY \*\* 2); // calculate distance between player

      //if the distance is less or eaqual to the enemies range

      if (dist <= i.range) {

         i.vx = (diffX / dist) \* i.speed; // calculate velocity in x-direction and store in enemy object

         i.vy = (diffY / dist) \* i.speed; // calculate velocity in y-direction and store in enemy object

         i.x -= i.vx; // update position

         i.y -= i.vy; // update position

         // if the enemy and player are overlapping

         if (checkOverlapp(i.x - i.w / 2, i.y - i.h / 2, i.w, i.h, player.relPos.x - player.w / 2, player.relPos.y - player.h / 2, player.w, player.h)) {

            damagePlayer(i.dmg); // deal damage to player

         }

      } else {

         // if out of range, stop movement of enemy

         i.vx = 0;

         i.vy = 0;

      }

   }

}

function damagePlayer(n) {

   // if the player has more or equal armor as damage dealt

   if (player.armor >= n) {

      player.armor -= n; // subtract the damage from the players armor

   } else {

      // if player has less armor than damage dealt

      n -= player.armor; // remove the armor value from the damage

      player.armor = 0; // set armor to 0

      player.health -= n; // substract the player health with the lasting damage

   }

}

Jag valde att lägga arrayen enemies i varje backgrundsobjekt. På så sätt går det lättare att byta mellan omgivningar, som då kan ha egna fiender.

Nu behöver jag bara inplementera att spelaren kan skjuta på fienden, och kanske ge fienden en healthbar så finns det faktiskt något att göra i spelet.

När fienden dör kan spelaren få pengar baserad på fiendens svårighetsgrad.

Så nu kan spelaren döda fienden. Jag lade till en for-loop i updateProjectilePosition() som med hjälp av checkOverlapp() kollar ifall projektilen överlappar fienden. Om den gör det så skadas fienden och projektilens objekt tas bort från arrayen projectiles.

I funktionen updateEnemies() kollar programmet ifall fiendens health är lägre eller lika med noll, om den är det så tas fienden bort.

Nu har fienden också en healthbar, och spelaren får in 20 pengar av att döda en fiende. Det enda som behövs nu är att lägga in fiender i kartan. Jag har hittils bara testat genom att använda createEnemy() i consolen.

Fiendera kan dock röra sig genom väggar. Men de kanske är spöken :O

Det är någonting jag komemr att fixa i framtiden iallafall. Det kommer väl vara att fienden ska finna en möjlig väg till spelaren. Där känner jag dock att det snabbt blir väldigt komplicerat. Jag får googla och undersöka tekniker för det när det kommer till det. Just nu är jag nöjd.

Jag kan ge en uppdaterad bild över main loopen:

function main() {

   checkCollision();

   checkBorders();

   checkWeaponFired();

   updateProjectilePosition();

   updateEnemies();

   updatePosition();

   checkInteraction();

   drawFrame();

   requestAnimationFrame(main);

}

**2022-01-02**

Igår kollade jag efter någon bra algoritm för att hitta närmaste spår från en punkt till en annan. Jag kom förbi A\* algoritmen men den kommer nog att vara på tok för långsam, om jag inte sätter ett upp ett rutnät eller punkter som är långt ifrån varandra. Fienden kommer då också behöva avgöra när den ska använda sig av A\* algoritmen och inte. Eftersom att det tar sådan lång tid tror jag nästan att det kan vara dags att använda sig av web workers. Huvudprogrammet kommer att be om ett spår från web workern, och kunna fortsätta köra medans algoritmen kör för fullt.

Jag har bara använt mig av web workers en gång, och det var i en övning, så det kan vara en utmaning. Jag har aldrig skrivit A\* algoritmen, men kollade på en timmes video igår från the Coding Train (väldigt bra) och lite pseudokod och det verkar inte vara helt omöjligt.

Innan jag börjar försöka implementera det i spelet borde jag nog testa att skriva en A\* algoritm för att öva. Det tänkar jag göra nu.

Jag lägger den i samma mapp som projektet, i en mapp som heter ”Test”, ifall du vill kolla på den.

Kort sagt är min uppfattning av A\* algoritmen är att man gör en grundad gissning på vilka alternativ som kan uteslutas. T.ex. kan man gissa på att gå bakåt inte är optimalt för att ta sig framåt. På så sätt så utesluter man många alternativ att ta sig till ett mål och det resulterar i att man mycket snabbare kan få resultatet.

Okej nu har jag kollat på videon igen och skrivit samma program som han skrivit. Jag har lärt mig mycket och förstår hur programmet fungerar. Nu ska jag kommentera hela programmet så jag verkligen förstår vad som händer, och sedan försöka få det att köra snabbare, och skriva ett för spelet. Just nu kan varje loop inte ens få plats i en frame… lite konstigt.

Okej nu har jag kommenterat hela med egna ord. Såhär fungerar det:

Programmet börjar med att kolla på startpunktens grannar. För varje granne, så länge den inte är bortlagt, räknar den ut en kostnad för att ta sig dit. Om den uträknade kostnaden är lägre än den kostnad den redan har, så byter den ut kostnaden och spar det spåret istället för det spår som grannen hade innan. Slutligen adderas kostnaden med återstående sträckan till målet, och den summan lagras i grannarnas f-värde. Detta loopas, och i varje loop väljs den punkt med lägsta f värde. Den punkt ses då också som färdig och läggs bort. På så sätt kommer de bästa punkterna att prioriteras. Genom att gå tillbaka till varje punkts ”förälder” så kommer bästa spåret att plockas fram.

Det låter som att den ska vara en relativt snabb algoritm, men ibland krävs det tre frames för att köra. Jag undrar om jag skrivit fel någonstans eller liknande… Men det är därför web worker skulle vara en bra idé. Nu ska jag ta bort allt onödigt hade jag tänkt för att göra det snabbare.

Okej jag fick ned tiden från max ca 900ms till 50ms. Det är fortfarande för långsamt för att köras varje frame så det kommer att ligga i en web worker.

Jag har läst på lite igen om webworkers och det verkar inte vara några konstigheter. Imorgon försöker jag implementera det i spelet.

Om jag kan uppdatera fiendens destination varje 50ms så kanske fienden kan använda sig av algoritmen alltid. Men det blir nog problem när det finns flera fiender.

**2023-01-03**

Fy vad jobbigt det här blir nu känner jag. Jag tror jag avstår från att implementera A\* algoritmen tills vidare. Men det var roligt att lära sig en ny algoritm!

**2023-01-05**

Igår så försökte jag animera fienderna när de rör på sig, men jag stötte på problem. När en fiende skapas så skapas den med en Construktor.

new CreateEnemyObject(1000, 1000, 50, 50, 1, 120, 3, 0, 200, enemyImg)

I funktionen / construktorn CreateEnemyObject() så försökte jag skapa en setInterval() loop som skulle loopa en metod i objektet. Metoden skulle ändra this.img.src för att skapa en animation. Men det verkar som att om man använder sig av this i en metod, så tappar den bort vad this refererar till.

Det gick iallafall inte.

Jag kommer nog komma tillbaka till det lite senare.

I framtiden kommer jag vilja dela upp programmet i flera filer, något jag inte gjort än.

**2023-01-12**

Idag på lektionen har jag börjat på en ny bild som kan öppnas vid ett tält i slottet:



Här kommer man kunna starta en ny strid för att tjäna pengar, stiderna kommer att bli svårare och svårare.

Den här bilden öppnas och fungerar på samma sätt som tidigare bok från **2022-12-28**.

Det är en if-sats i drawFrame() funktionen som kollar ifall bilden ska vara öppen, och om den är det så ritas bilden:

   if (fightBookImg.isOpen) ctx.drawImage(fightBookImg, fightBookImg.xPos, fightBookImg.yPos);

Booleonen fightBookImg.isOpen sätts till true tidigare i programmet, om spelaren trycker på tangenten ”e” samtidigt som spelaren befinner sig på korrekt plats.

Jag kom precis på att om capslock är aktiverat så reagerar inte programmet på någon av tangentryckningarna. Detta för att t.ex. ”e” != ”E”. De har inte heller samma keycode. Istället löser jag det med event.key.toLowerCase(). Nu är det fixat.

**2023-01-13**

Nu ser senaste bilden ut såhär:



Nu ska jag skapa en bild för varje siffra. På så sätt kommer jag kunna skriva vilken level och pris. Jag skulle kunna göra det med javascript i canvasen med ctx.fillText() men jag vill att det ska ha samma stil som resten av den ritade texten.

Här är sifforna jag ritat:



Jag skapar en array som lagrar alla bilder på sifforna:

let mediumNumbers = [];

for (let i = 0; i < 10; i++) {

   let img = new Image();

   img.src = "img/mediumNumbers/num" + i + ".png";

   mediumNumbers.push(img);

}

Bilderna kommer att ligga i arrayen i lämplig ordning. mediumNumbers[2] kommer att innehålla siffran 2 osv.

Nu har jag också skrivit en funktion drawNumbers() som ritar ett tal i önskad position. Här är funktionen:

function drawNumbers(x, y, num, t) {

   num = String(num); // Convert the num to string data type

   let xPos = x; // Set first xPos to x

   // If the desired type is "medium"

   if (t == "medium") {

      let space = 5; // Space between numbers

      // then for every digit

      for (let i of num) {

         ctx.drawImage(mediumNumbers[parseInt(i)], xPos, y); // draw it's corresponding num

         xPos += mediumNumbers[parseInt(i)].width + space; // update xPosition by adding width and space

      }

   }

}

Jag är nöjd med funktionen, det funkar väldigt bra. Jag behövde göra om talet till en sträng för att kunna stega igenom varje siffra. Då behöver jag också konvertera tillbaka siffran till ett tal när jag sedan använder det som index i mediumNumbers[].

Funktionen kallas sedan på i drawFrame() om boken är öppen:

if (fightBookImg.isOpen) {

      ctx.drawImage(fightBookImg, fightBookImg.xPos, fightBookImg.yPos); // draw book

      drawNumbers(fightBookImg.xPos + 245, fightBookImg.yPos + 275, fightBookImg.level, "medium"); // draw level number

      drawNumbers(fightBookImg.xPos + 235, fightBookImg.yPos + 355, fightBookImg.price, "medium"); // draw price number

   }

Såhär ser det ut i spelet:



Mycket nöjd.

Jag väntar lite med att faktiskt göra levlarna. Nu tänker jag skapa ett inventory, ett lager för föremål.

**2023-01-14**

Just nu tänker jag bygga ut drawNumbers() funktionen så att den kan rita mindre bokstäver också. De kommer bland annat att användas i boken där man kan uppgradera sin gubbe.

Jag börjar med att skapa mindre siffror i GIMP.

Jag ritade bilderna, och lagrar de i början av programmet i en array som heter smallNumbers. I drawNumbers() skrev ut en till if-sats i som kollade ifall önskade storleken är ”small” eller ”medium”, och gjorde på samma sätt som med ”medium” storleken. Sedan lade jag till en till parameter, sign. Tanken är att kunna rita symboler, som t.ex. en pil.

Jag ritade en pil, och lagrade den i ett objekt, signs. Varför jag väljer att lägra symbolerna i ett objekt och siffrorna i en array är för att siffrorna är lätta att lägga i en viss ordning som överensstämmer med sifforna i sig. Med symbolerna vill jag istället plocka ut en viss symbol genom att ange dess namn. Nu når jag pilen med signs.arrow vilket är mycket smidigare.

Funktionen drawNumbers() kan nu alltså rita ett tal, och sedan en symbol efter det. Men om jag vill rita ett tal, symbol och sedan tal igen, valde jag att i slutet av funtkionen returnera nästa x position. Då för att skriva tal, symbol, tal kan jag göra såhär:

let xPos = drawNumbers(x,y, tal, storlek, “arrow”); // Här ritas ett tal med en pil, och lagrar nästa x

drawNumbers(xPos, y, tal, storlek)

Nu kommer de att ritas efter varandra. Vilket t.ex. kan skriva 100 => 110, något som jag använder här:

En bild som visar text, klocka

Automatiskt genererad beskrivning

Jag märkte också att helt plötsigt så centreras inte böckerna när de öppnas i spelet. Jag misstänkte att det hade att göra med att deras bredd och höjd inte hann läsas in innan de användes i följande beräkningar:

   armsBookImg.xPos = midWidth - armsBookImg.width / 2;

   armsBookImg.yPos = midHeight - armsBookImg.height / 2;

Där midWidth och midHeight är halva canvasens bredd och höjd.

Det resulterade i att böckernas hörn befann sig i mitten av skärmen, istället för böckernas mitt. Jag googlade på en lösning och hittade denna lösning:

let armsBookImg = new Image();

armsBookImg.onload = function () {

   armsBookImg.xPos = midWidth - armsBookImg.width / 2;

   armsBookImg.yPos = midHeight - armsBookImg.height / 2;

};

armsBookImg.src = "img/books/ArmsBook.png";

När bilden har laddat in så genomförs beräkningen. Nu blir böckerna centrerade igen.

**2023-01-18**

Jag har lagt till levlar. Jag valde att strukturera det på så sätt att fight-bok objektet innehåller en till array, som i sin tur innehåller ett objekt för varje level. fightBookImg.level[0] ger alltså objektet för första leveln.

Objeken innehåller i sin tur antal fiender, hur mycket skada de ska kunna motstå, hur snabbt de ska skada spelare, hur snabbt de ska röra på sig, vid vilket avstånd de ska jaga spelaren samt ett pris för att avklara leveln. Några exempel:

fightBookImg.level = [

   { enemyAmount: 2, enemyHealth: 50, enemyDmg: 0.1, enemySpeed: 2, enemyRange: 400, reward: 100 },

   { enemyAmount: 4, enemyHealth: 75, enemyDmg: 0.2, enemySpeed: 2, enemyRange: 700, reward: 150 },

För att starta leveln har jag också skapat en metod i fightBookImg. Varför jag valde att göra en metod och inte en funktion, är för att jag i framtiden kanske vill kunna ha flera levlar på olika ställen. Nu startar jag en level i slottet genom att skriva fightBookImg.startLevel(); , på så sätt blir det lättare om jag sedan t.ex. vill skriva castleImg.startLevel();. För att hålla koll på vilken level spelaren befinner sig på, så lade jag till ännu ett värde i objektet som heter currentLevel. Den används i metoden startLevel() för att starta den level som spelaren inte avklarat än.

I startLevel() skapas nya fiende objekt i castleImg.enemies[] genom att använda sig av konstruktorn CreateEnemyObject som jag skapat tidigare i utvecklingen.

Metoden sätter också en booleon, fightBookImg.levelStarted till true om en level har startats.

I main() loopen lade jag in en funktion som heter checkGameRules(). Där kollas just nu bara ifall en level har startats, och om antal fiender på den kartan är noll. Om det är det, så innebär det att spelaren har bekämpat alla fiender, och leveln är avklarad. Då kallar programmet på en annan metod i fightBookImg som heter finishLevel(). Den metoden sätter fightBookImg.levelStarted till false, ger spelaren pengar och uppdaterar fightBookImg.currentLevel.

Här är de två metoderna:

/\* This method starts the next level \*/

fightBookImg.startLevel = function () {

   let lvl = this.level[this.currentLevel]; // For simplicity, create and set lvl to the current level object

   this.levelStarted = true; // A level is started, set to true

   // For every enemy objekt

   for (let i = 0; i < lvl.enemyAmount; i++) {

      let x = 1500 + Math.random() \* 1200; // Generate random x-position

      let y = 150 + Math.random() \* 1710; // Generate random y-position

      castleImg.enemies.push(new CreateEnemyObject(x, y, 50, 50, lvl.enemyDmg, lvl.enemyHealth, lvl.enemySpeed, 0, lvl.enemyRange)); // Push a new enemy object into the castleImg.enemies array

   }

};

/\* This method finished the current level,

   and updates it to the next level \*/

fightBookImg.finishLevel = function () {

   let lvl = this.level[this.currentLevel]; // Create lvl and set it to the current level object

   this.levelStarted = false; // Set levelStarted to false

   player.money += lvl.reward; // Give the player it's reward

   this.currentLevel++; // Update current level with 1

};

”this” refererar då till fightBookImg.

Nu ska jag lägga till så att spelaren faktiskt kan ”dö”.

**2023-01-24**

Idag har jag kommenterat min kod och lagt till lite lådor i slottet.

**2023-01-26**

Idag har jag ritat mer på upgradering boken, och lagt till att spelarens vapen har en maximal räckvidd. Det är något jag börjat göra så att man kan uppgradera. Jag har också ritat ett nytt tecken.

Nu när jag kom hem ritade jag även lite på en karta för grottan, och lade till ett filter när man är i grottan som gör att man får mindre synfält. Såhär ser det ut:

En bild som visar pil

Automatiskt genererad beskrivning

**2023-01-31**

På lektionen idag påbörjade jag att lägga till information i spelet, så att spelaren vet vad som ska göras. Jag ritade en bild med ett ”i” som man ska klicka på i spelet, men jag fick ett problem att programmet krävde information om bilden innan den hade laddats in. Jag bestämde mig för att lösa detta för gott genom att programmet ska ladda in alla bilder, och vänta på att alla är inladdade, innan spelet börjar. Jag pratade med Mauritz hur man kan få programmet att ”vänta” på bästa sätt, och han föreslog att jag använder mig av ytterligare en requestanimationframe loop, som inte kör samtidigt som min huvudloop.

Jag skapade en ny uppsättning filer, eftersom att detta kommer innebära stora förändringar för programmet.

Till min förvåning så lyckades jag få det att fungera direkt. Nu laddas alla 35 bilder in, innan de används någonannanstans i programmet. Det krävde en del omstrukturering av programmets början.

**2023-02-01**

Jag har nu gjort en informationssida som man får upp genom att klicka på en symbol nere i vänstra hörnet. Först skrev jag texten i programmet, genom ctx.fillText(). I den metoden så anger man en max width. Men tyvärr så ”wrappas” inte texten om den överskrider max bredden. Istället trycks texten ihop. Jag googlade på hur man kan få texten att wrappa, men det verkade inte finnas något snyggt sätt. Eftersom att texten inte kommer att ändras, utan vara konstant så valde jag att skriva det i bildprogrammet istället.

För att testa hur krävande programmet är just nu, så skrev jag ut tiden det tar för att en hel mainloop ska köras. Om den tiden skulle överskrida 16 ms så börjar det lagga. Eftersom att varje karta/omgivning innehåller olika antal objekt osv, så kommer prestationen att variera mellan dem. Hittils är grottan den mest krävande kartan, eftersom att den har flest kollisionsobjekt. Där snittar tiden för en frame på ca 1 ms, och toppar på 1,5 ms. Det känns mycket tryggt. Jag förväntade mig att programmet skulle vara lite långsammare.

Lite inför framtiden:

”Städa upp” första omgivningen, och rita färdigt den.

Lägga in alla kollisionsobjekt för grottan ( Just lagt in ca 50 objekt).

Göra något utav grottan.

Skapa en ny variant av fiende.

Lyckas animera fienderna när de rör på sig .

Göra så att programmet kollar framerate, och anpassar efter det.

Göra en startskärm.

Göra färdigt uppgraderingsboken.

Göra så att spelaren kan köpa saker, typ hälsodrycker osv, och kan ha det i sitt inventory.

Om jag hinner:

Göra så att spelarens framsteg lagras lokalt.

**2023-02-02**

Idag gjorde jag en till requestanimationloop, som kallas på när den första requestanimationloopen är färdig och inte loopas längre. Den nya loopen kör 10 gånger, och kollar vilket framerate datorn har, eller snarare hur lång tid en frame tar. Detta används sedan för att räkna ut en ”speedConstant” som jag kallar det. Eftersom att jag har utgått från en frameRate på 60, dvs en tid på 16,6666 ms, så kommer den frameRaten att ge speedConstant 1. Formeln blir då k = frameTime / (1000/60) . Om spelaren har lägre frameTime, dvs högre framRate, så kommer spelaren alla hastigheter att bli mindre, och tvärtom.

**2023-02-03**

Idag gjorde jag en laddskärm. Den har en förloppsindikator som just nu bara visar hur stor andel av bilderna som laddats in. Sedan kör en request animationframe som räknar ut en speedConstant, och sedan är spelet igång.

**2023-02-03**

Idag funderar jag på att ge mig på att skapa en ny fiende. Den nya fienden ska ha ett annat utseende och storlek, men ska också kunna skjuta mot spelaren.

Nu fungerar den nya fienden.

Jag skapade en ny array för att separera fiendens projektiler med spelarens, och lite nya parametrar i olika funktioner för att kunna bygga upp olika objekt i samma konstruktor, beroende på vilket fiende det är.

Jag skapade en metod åt fiende objeket, som skapar en projektil med uträknade hastigheter osv. I konstruktorn skapar jag ytterligare två metoder, en som startar en setInterval loop och en som stannar den. Såhär ser det ut:

      this.shoot = function () {

         let dist = Math.sqrt((player.relPos.x - enemyObject.x) \*\* 2 + (player.relPos.y - enemyObject.y) \*\* 2);

         let vx = -(enemyObject.x - player.relPos.x) / dist;

         let vy = -(enemyObject.y - player.relPos.y) / dist;

         enemyProjectiles.push(new CreateProjectile(enemyObject.x, enemyObject.y, vx, vy, 10));

      };

      this.shootInterval = undefined;

      this.startShoot = function () {

         enemyObject.shootInterval = setInterval(enemyObject.shoot, 200);

      };

      this.stopShoot = function () {

         clearInterval(enemyObject.shootInterval);

      };

Jag stötte på ett problem. Tidigare, där det står ”enemyObject” så stod det istället ”this”. Jag märkte dock att i det läget så refererade inte ”this” till fiende objektet, utan till objektet window. Därför valde jag att i början av konstruktor funktionen skriva: let enemyObject = this;

På så sätt kunde jag fortsätta referera till fiende objektet.

Projektilerna uppdateras i samma funktion som spelarens projektiler, updateProjectilePosition(), fast i en annan for-loop. Det vore egentligen snyggt att ha dem i samma for-loop, men eftersom att de är i två olika arrayer och behandlas olika så ser jag inte en övervägande poäng med det.

Projektilerna ritas på samma sätt som spelarens projektiler, i en for-loop i drawFrame() funktionen.

**2023-02-05**

Jag upptäckte en bugg med förra fiendens projektiler. Om fienden sköt, och ett senare skott träffade något, så togs alla före detta skjutna projektiler bort. Jag löste det genom att göra två saker. Det första jag gjorde var att ändra i splice() metoden. Där hade jag skrivit splice(i, 1), då i syftar på det objekt som ska tas bort, inte vilken index i arrayen. Så i for-loopen som detta sker, ändrade jag i till arrayens längd, istället för att skriva en ”for( let i of…”. Det jag också gjorde va att gå igenom arrayen baklänges. Detta gjorde jag för att metoden splice() ändrar arrayens längd, vilket resulterar i att en projektil blir överhoppad om den tidigare projektilen träffat något. Det va inte märkbart i programmet, men nu när jag ändå höll på och ändra i funktionen, så passade jag på att göra denna ändring. Eftersom att det är exakt samma procedur för spelarens projektiler, så gjorde jag samma ändringar där, även om det inte märks i programmet.

Jag lade lite snabbt också till att spelaren kan se exakt hur mycket hälsa som är kvar, och vilken som är spelarens maximala hälsa. Detta för att spelaren tydligare ska se vad som händer när hälsan uppgraderas vid smeden.

**2023-02-07**

Igår lade jag till att när spelaren lägger musknappen på något som man kan klicka på, så visas det vad det är för något. T.ex. om man lägger musknappen på informations knappen så säger det ”information” osv. Jag gjorde detta genom att skriva en funktion   
drawMessageAtMouse(text, maxWidth). Jag hade utforskat metoden ctx.fillText() tidigare, och den har en parameter som heter max width. MEN, istället för att ”wrappa” texten om den överskrider bredden, så krymper den ihop den på längden. Här är första gången jag rent av kopierar någon annans kod. Det här är koden jag kopierade:

let words = text.split(" "); // Create an array containing all words in the text

   let lines = []; // This array will contain content of all lines

   let currentLine = words[0]; // First line will start with first word

   // for every word, except the first word

   for (let i = 1; i < words.length; i++) {

      let word = words[i];

      let width = ctx.measureText(currentLine + " " + word).width; // Messure the next width

      // If the next width is less than the max width, add the word to the current line

      if (width < maxWidth) {

         currentLine += " " + word;

      } else {

         // If the next width is larger than max width, store the current line, and start the next line with the current word

         lines.push(currentLine);

         currentLine = word;

      }

   }

   lines.push(currentLine); // Finally, store the last line in the lines array

Kommentarerna fanns inte, de skrev jag dit. Jag förstår hur det fungerar, och lyckas integrera det i min funktion. Nu har jag alltså en array, lines[], som innehåller alla rader text. Här skriver jag ut texten senare i funktionen, (detta har jag skrivit själv):

let x = mouse.x + 20;

   let y = mouse.y + 20;

   let rectW = lines.length > 1 ? maxWidth : ctx.measureText(currentLine).width + 10;

   ctx.fillStyle = "rgba(0,0,0,0.5)";

   ctx.fillRect(x, y, rectW, lines.length \* 20);

   ctx.textAlign = "left";

   for (let i in lines) {

      let line = lines[i];

      ctx.fillStyle = "white";

      ctx.fillText(line, x + 5, y + i \* 20 + 15);

   }

Ovanstående funktion kallas på varje frame i drawFrame() funktionen, om det finns ett meddelande. Medelandet lagras i objektet mouse, och uppdateras varje gång musen rör på sig med en eventlistener.

­­­**2023-02-08**

Idag lade jag till så att spelaren kan ändra storleken på skärmen. Det va svårt att få till så att programmet uppdaterade positionen på alla bilder rätt. Jag skapade två nya funktioner för det här. En funktion kallas på av en eventlistener som sitter på window objektet. Den funktionen ändrar några värden, och kallar på funktionen configureImages() som configurerar alla bilder igen. I slutet av den funktionen kallas en ny funktion, placeImages() som uppdaterar bildernas x och y position, om de är placerade relativt till något. Om de är det, så har de ett värde som heter subXPos och/eller subYPos. Dessa kan antingen sättas till ett värde, då placeras de ett angivet avstånd från canvasens botten eller högra sida. Eller så kan man sätta värdet till strängen ”center” och då centrerar placeImages() den. Här är de två funktionerna:

function resizeWindow() {

   canvas.width = window.innerWidth;

   canvas.height = window.innerHeight;

   midWidth = canvas.width / 2;

   midHeight = canvas.height / 2;

   configreImages();

   moveToRelPos(player.relPos.x, player.relPos.y);

}

function placeImages() {

   for (let i of allImages) {

      if (i.subXPos) {

         if (i.subXPos == "center") {

            i.xPos = midWidth - i.width / 2;

         } else {

            i.xPos = canvas.width - i.subXPos;

         }

      }

      if (i.subYPos) {

         if (i.subYPos == "center") {

            i.yPos = midHeight - i.height / 2;

         } else {

            i.yPos = canvas.height - i.subYPos;

         }

      }

   }

}

**2023-02-09**

Jag tror jag va lite trött igår. Idag kom jag på att jag inte behöver kalla på configureImages() funktionen. Det löste en del problem, t.ex. att om man startade en level, och sedan ändrade storlek på skärmen så försvann alla fiender. Det beror på att i configureImages() så sätts arrayen enemies[] till en tom array. Men nu är det löst.

Nu kom jag på varför jag behövde kalla på funktionen. På vissa ställen i configureImages() använder jag mig av bilders x och y koordinater. De rader som gör det lägger jag i en ny funktion refreshImageLocation(). Den kallas på i configureImages() och i resizeWindow(). Nu funkar det.

På lektionen gjorde jag en animation för den ena typen av fiende. Jag och Mauritz kollade länge på min konstruktor för att försöka hitta varför en variabel vägrade sätta sig till false. Efter ett långt sökande visade det sig bara vara ett stavfel…

Nu senare på kvällen gjorde jag om lite på spelarens amination också. Tidigare så kunde spelaren flimra lite. Det berodde på att jag animerade på så sätt att jag bytte source, URL, på spelarens bild. Det resulterade i att bilderna behövde ladda in på nytt varje gång. Nu har jag istället gjort så att spelarens bild, byter till en redan inladdad bild. Dvs jag laddar in alla bilder i början av programmet, precis som för alla andra bilder, och byter mellan dem för att animera.

// Create an array containing all player images

let playerImages = []; // Create global array

// As many times as amount of images

for (let i = 1; i <= 12; i++) {

   let img = new Image(); // Create new image

   //Assign method when the image is loaded, this will be used in loading screen

   img.onload = function () {

      img.isLoaded = true;

   };

   img.src = "img/WalkAnimation/PlayerWalk" + i + ".png"; // Give its URL

   playerImages.push(img); // Place the new image object in the array

   allImages.push(img); // Also place it in the array containing all images

}

**2023-02-10**

Idag ska jag göra en pausmeny. Det kommer möjliggöra några nya funktioner. T.ex. att stänga av musiken, spara osv.

**2023-02-25**

Idag blev jag färdig (tror jag) med sparfunktionen. Klickar spelaren escape och sedan Save, så kommer data sparas lokalt. Jag tror jag sparar allt det viktiga. Jag sparar t.ex. spelarens pengar, position, vilka fiender som finns i världen och deras hälsa osv. Jag sparar också spelarens uppgraderingar. Något jag inte sparar är om man har öppnat en ”bok”, men jag tycker inte det känns viktigt.

Jag stötte på några problem då och då och behövde ta pauser för att inte köra fast. Det första jag stötte på var att jag gärna ville spara hela currentEnv objektet, dvs det objekt som innehåller all information och nuvarande karta. För att spara ett objekt i localStorage så behöver man göra om det till en sträng. Det gör jag genom metoden JSON.stringify(). Det blev ett klurigt felmeddelande. Jag misstänkte att det hade att göra med att jag försökte spara en bild, men efter att jag och Mauritz kollat på felmeddelande så hade det att göra med att objektet refererade till ett objekt, som refererade tillbaka till första objektet. Det bildade en loop och kunde inte skrivas i form av sträng. Istället så fick jag spara namnet på objektet, och sedan med hjälp av det sätta till rätt omgivning när spelet laddade in.

De andra problemen jag stötte på var lite liknande. Jag kunde inte spara alla fiendeobjekt och sedan lägga tillbaka dem när spelet laddades in. Där hade det att göra med att jag försökte spara en bild lokalt, eftersom att fiendens bild är lagrad i fiendens objekt. Då fick jag istället göra en loop som för varje sparad fiende skapade en likadan fiende med CreateEnemyObject().

Nästa sak att göra är kanske att lägga in en inställnings meny, och kanske att göra en funktion som visar meddelanden en kort stund. Den kan utnyttjas för att t.ex. säkerställa spelaren att den har sparat spelet, genom att det kommer upp ”Saved”. Programmet gick precis över 2000 rader kod.

Förresten, de funktioner som sköter sparandet är saveGame(), loadSave() och clearSave(). Du hittar dem längst ned i programmet. loadSave() körs varje gång programmet startas, efter att programmet hämtat frame raten.

Så, nu har jag också skrivit funktionen displayMessage() som ritar ett meddelande under en given tid. Den används just nu bara för att skriva ”Game Saved!”

**2023-02-26**

Idag lade jag till en inställnings sida som man kan öppna genom att klicka på escape och sedan settings. Hittils har sidan inte någon verklig funktion, men man kan checka i auto save och musik. Ännu har det ingen funktion. För att göra en autosave tänker jag helt enkelt skapa en setInterval loop som kallar på funktionen saveGame().

Jag vill också ha en knapp som kallar på funktionen clearSave(). Den kan förslagsvis också finnas på settings.

**2023-02-27**

Jag har gjort om funktionen checkClickables(). Förut bestod den till stor del av if-satser som kollade vilken tagg ytan som spelaren klickat på har, och gjorde olika saker efter det. Nu har jag istället inkluderat en metod onClick() i varje clickable. Då behöver inte programmet kolla efter tagg osv, utan kör bara ytans metod onClick. Mycket snyggare och effektivare.

**2023-03-01**

Idag har jag lagt in en del ljudeffekter: När spelaren öppnar och stänger en bok, när spelaren skjuter och fem olika ljud för när spelaren tar skada. Jag bytte även bakgrundsmusik till enligt mig en mycket passande hobbit låt.

Det va kul!

Jag lärde mig också att om jag vill spela samma ljud samtidigt så behöver man klona ljudet. Det görs enkelt med att skriva ljud.cloneNode().play(); Så kan ljudet spelas hur mycket som helst. Det är användbart när spelaren skjuter.

**2023-03-02**

Jag stötte på ett problem. Det visade sig att när jag använder ljud.cloneNode() så återställs t.ex. ljud.volume. Det gjorde så att jag inte kunde bestämma ljudnivå på ljudet.

För att lösa detta gjorde jag en funktion som heter playClone(). Såhär ser det ut:

function playClone(sound) {

   let s = sound.cloneNode();

   s.volume = sound.volume;

   s.play();

}

Den skapar en variabel ”s” och ger en klon av ljudet till den. Sedan ändrar jag volymen på klonen till orginalljudets bestämda volym, och sedan spelar jag upp klonen.

**2023-03-09**

Senaste dagarna har jag dels ritat en ”bok” för att kunna köpa saker i affären i huset. Jag lade in det i programmet och mappade ut var spelaren kan trycka osv. Sedan skrev jag två metoder i spelarobjektet. En metod heter addToInventory() och lägger till ett givet objekt till inventoryt. Den andra metoden heter structureInventory() som placerar ut alla objekt i inventoryt. Detta är användbart för när t.ex. spelaren använder ett objekt i inventoryt, så försvinner det objektet och då vill man strukturera om så att alla objekt ligger bredvid varandra.

Nu har jag också gjort så att inventoryt sparas när spelet sparas och laddas in när spelet startas. Det gör jag på samma sätt som med fienderna. Jag kan inte spara player.inventory och sedan sätta in hela den arrayen när programmet öppnas, för att arrayen innehåller objekt som innehåller bilder. Det blir problem. Så istället gör jag en for-loop som för varje objekt i player.inventory skapar ett nytt objekt med CreateItem(). Se funktionen loadSave();

Reflektion

Jag är nöjd med programmet. Jag upplever att det var som svårast i början, att skapa en grund som fungerade bra, t.ex. att bra kollision mellan spelaren och väggar, eller en smidig övergång mellan att flytta bakgrundsbilden till att flytta spelaren över skärmen.

När det var etablerat så var programmet väldigt flexibelt. Jag kunde enkelt lägga in nya objekt innehållande position och dimensioner för nya kollisionsobjekt, som sedan hanterades automatiskt i programmet. Delar av den koden kunde jag använda i andra syften, som t.ex. kolla ifall spelaren har interagerat (tryckt på tangenten E) med ett objekt i närheten av spelaren, eller för att skada spelaren eller fiender vid kollision med projektiler.

Jag hade större ambitioner kring de fiender som finns just nu. Det finns två typer av fiender, en fiende jagar spelaren den kortaste vägen och skadar spelaren om de överlappar varandra, och den andra fienden har samma funktion, men går långsammare samtidigt som den skjuter mot spelaren. Jag experimenterade med att använda mig av A\* algoritmen för att fienden ska kunna beräkna kortaste vägen till spelaren runt hinder, men jag märkte att den algoritmen var alldeles för långsam. Jag insåg att det inte är så viktigt med komplicerade fiender, och nöjde mig med att de rörde sig på ett enkelt vis.

Jag ägnade också tid åt att göra programmet robust. Det första jag gjorde var att räkna ut uppdateringshastigheten på webbläsaren, för att därefter ändra hastighet på spelaren, fiender, samt projektiler för att programmet ska kännas likadant på alla datorer. En annan sak jag gjorde för att göra programmet robust var att tillåta spelaren att ändra storlek på sitt fönster. Det krävde en del omskrivning av koden, men med två nya funktioner gick det att lösa. Jag skrev även om programmet så att alla bilder laddades in i början, och skapade en skärm med en förloppsindikator som visar hur många bilder som laddats in, och processen att räkna ut frame rate.

Det sista stora momentet var att gör det möjligt att spara framgångar lokalt, enkla inställningar som att stänga av bakgrundsmusik och sätta på eller av autosave. Det gick förvånansvärt bra, efter att ha bråkat lite med JSON.stringify() och localStorage.

Vad är jag mindre nöjd med och kan göras bättre?

I programmet finns det delar som repeterar samma process, fast med variationer i tecken och variabler som inte gör det självklart hur man kan göra en funktion av det hela, för att undvika repetition. Det blev också väldigt många objekt, vissa som behövde loopas igenom varje frame (kollisions objekt). Det uppstod inga prestationsproblem, jag testade hur lång tid en frame tog när jag belastade den så hårt jag kunde, och nådde en tid på ca. 1,6 ms av 16ms tillgängliga. Som förebyggande åtgärd för framtida problem med prestatanda kan det vara lämpligt att skriva programmet så att det bara kollar på de kollisionsobjekt som är nära spelaren. Ett annat sätt jag kan tänka mig, fast inte så robust kanske, är att ge kollisionsobjekt en viss färg. Om spelaren är på väg in i ett t.ex. svart område, så tillåts det ej. Bara en idé.

Fortsatt utveckling av programmet kan vara att ge grottan ett syfte, hittills kan inte spelaren göra något inne i grottan, men den visar iallafall potential till vad spelet kan utvecklas till och vad som är möjligt. Det kan också finnas något slutligt mål, som ger spelaren upplevelsen att den åstadkommit något. Man kan alltid utveckla fienderna till att ha kollision och göra flera varianter, eller en minikarta nere i hörnet osv. Man kan utveckla det hur långt som helst, och det är därför jag valde att stanna här, för egentligen finns det inte något naturligt stopp.

Min egna åsikt är att det vore schysst att dela upp programmet i flera filer, bara för att lättare kunna hitta delar av koden. Koden är nu 2500 rader, och det blir en del att leta i om man inte är bekant med koden. Däremot är det lättare att följa koden steg för steg när allt är i samma fil.

Något jag har lärt mig är hur förvånansvärt snabb datorn är, och hur mycket beräkningar den hinner med på mindre än en sextiondels sekund. Bara det påvisar datorns kapacitet till att lösa stora problem. Jag har också lärt mig en del om dess begränsningar, t.ex. genom att A\* algoritmen var för långsam. Jag har på så sätt insett vikten av genomtänkt, optimerad kod och smart kod för att inte nå datorns begränsningar.

I stort är jag väldigt nöjd med mitt program, inte minst med det estetiska, då jag ritat alla 50 bilder själv. Jag är också särskilt nöjd med hur jag lyckades integrera bilderna på ett väldigt naturligt sätt, och hur ljudeffekterna i spelet fick allt att kännas lite mer levande.

/Benjamin Simon Na20C