OPPGAVE 1 SNAKE

Dette er den oppgaven som er mest egnet til grupper med en til to kandidater.

BESKRIVELSE

Det skal programmeres et spill som heter «snake». Spillet handler om en «slange» som beveger seg over skjermen med en gitt fart. Det er ikke mulig å stoppe slangen i å bevege seg. Men den kan styres opp, ned, venstre og høyre. Tilfeldig rundt på brettet kommer det mat til slangen som slangen skal spise. Hver gang slangen spiser så blir den lengre og lengre. Slangen kan ikke kollidere med seg selv eller veggene i spillet, da dør den.

Spillet kan pauses med Space- tasten på tastaturet, og startes igjen ved å trykke på «Resume» knappen.

youtube: https://www.youtube.com/watch?v=hAz4bGzXPCc
wikipedia: https://en.wikipedia.org/wiki/Snake_(video_game_genre)
Tutorial: https://rembound.com/articles/creating-a-snake-game-tutorial-with-html5

ELLEMETER SOM SKAL VÆRE MED

a) Programmet skal ha et HTML definert canvas. Canvasets størrelse kan fastsettes i HTML delen, men eventuelle eventer skal programmeres i JavaScript.

b) Det skal vises poeng heletiden under spillet, og poengene er avhengig av hvor lang tid det har tatt fra en matbit dukket opp til slangen har spist den. Jo kortere tid jo høyre poengsum. Slangens fart skal øke gradvis etter som den konsumerer mat.

c) Brukeren skal ha mulighet til å start, stoppe og pause spillet. Samt start et nytt spill uten å oppdatere nettleseren.

d) Programmet skal ha globale variable, objekter og funksjoner. Funksjonsdefinerte objekter (Klasser), klassene skal ha både «Private» og «Public» medlemmer.

e) Programmet må inneholde enumererte statuser, løkker og logiske sjekker.

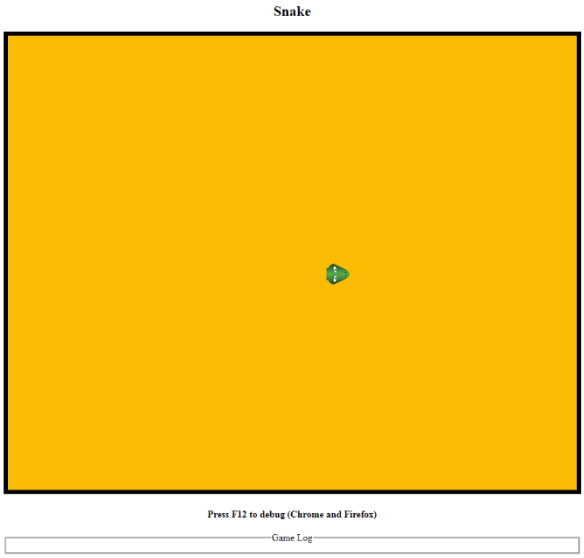
f) Klasse-struktur og navngivning står kandidaten/gruppen fritt til å velge selv, men det er viktig at notasjon og navngivning er entydig til objektets attributter og metoder. Samt alle navngitte variabler og konstanter skal være på engelsk.

g) Slangens visuelle elementer skal bestå av sprites som lastes via et sprite-sheet.

Det mest gunstige er å dele spillebrettet inn i rader og kolonner. En slange kropp er 38 piksler bred og høy. Det medfølger i koden et objekt som holder på antall rader og kolonner spillebrettet består av.

TASK 1 — 10 POENG

a) Hode til slangen:



Lag en funksjonsbasert klasse for eksempel TSnakeHead. La denne klassen ta inn to parametere slik at du får plassert hodet på riktig rad og kolonne.

La denne klassen ha en tegne funksjon slik at du kan tegne hodet ved hjelp av «sprite»-biblioteket som medfølger i koden du har fått.

Lag de nødvendige lokale medlemmene i klassen for «posisjon» «spriteinformasjon» og selve «Spriten».

Lag en global forekomst av TSnakeHead og opprett innholdet i «newGame» funksjonen før du påroper drawGame.

Videre så tegner du hodet i drawGame funksjonen.

b) Flytte hodet til slangen:

Videre så lager du en animasjon som flytter hodet ved hjelp av:

```
hndUpdate = window.setInterval(drawGame, 300);
```

Plasser denne koden i «newGame» funksjonen og oppdater «drawGame» slik at du får renset «Canvas» for hver oppdatering.

Utvid klassen TSnakeHead med to nye funksjoner. En som kan sette retning og en som kan flytte hodet. Du kan kalde disse for eksempel for «setDirection» og «move».

I «move» funksjonen så trenger du bare å gjøre en «switch / case» på retningen. Lag et globalt enum-objekt med retningen til slangen, for eksempel:

```
const EDirection = {Up: 0, Right: 1, Left: 2, Down: 3};
```

Lag nå en lokal variabel som du setter til oppstartsretning:

```
let direction = EDirection.Right;
```

Så kan du flytte hodet til slangen med «col++» eller «col--» og samme for «row».
I setDirection funksjonen kan du helt enkelt bare sette retningen:

```
direction = aDirection;
```

I «move» funksjonen må du nå flytte hodet så mange piksler som opptar en slangekropp. Du kan enten øke/minke x eller y med antall piksler, eller sette eksakt posisjon ved å bruke kolonne og rad informasjonen.

c) Riktig retning på hodet:

Legg så inn i tegne funksjonen til hodet riktig sprite i forhold til retningen på slangen. Du kan sette retningen på hodet ved hjelp av «setIndex» funksjonen til «sptiten»

I «cvsKeydown» funksjonen legger du inn nødvendig kode for å gi slangehodet ny retning når brukeren trykker på piltastene.

d) Game Over status:

Innfør nå game over status lik at når hodet treffer en av kantene i spillet så stopper spillet. Dette kan du gjøre i tegne funksjonen til hodet, her kan du sjekke posisjonen til hodet og sette spillet i «gameOver» modus. Husk å bruke enum-objekt når du jobber med programstatus.

Videre i «drawGame» funksjonen så burde du dele opp slik at du flytter før du tegner, da kan du stoppe spillet uten at hodet til slangen forsvinner.
Her er et lite hint på hvordan «drawGame» og «newGame» kan se ut:

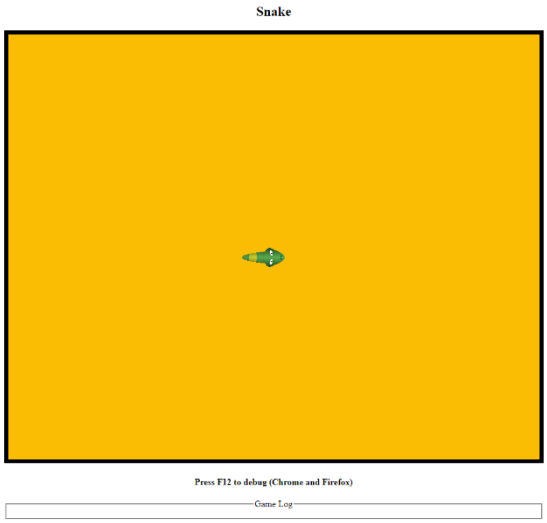
```
function newGame() {
    snakeHead = new TSnakeHead(10, 5);
    hndUpdate = window.setInterval(drawGame, 300);
    drawGame();
}
//-------------------------------------------------------

function drawGame() {
    snakeHead.move();
    if (gameStatus === EGameStatus.GameOver) {
        window.clearInterval(hndUpdate);
        addLogText("Game Over!");
        return;
    }
    ctx.clearRect(0, 0, cvs.width, cvs.height);
    snakeHead.draw();
}
//-------------------------------------------------------
```

Test spillet å se at du får endret retning på hodet med piltastene samtidig som at hodet har riktig retning. Test også at spillet stopper når det treffer en av kantene.

TASK 2 — 10 POENG

Vi legger nå inn halen til slangen.



a) Halen til slangen:

Start med å ta en fullstendig kopi av klassen «TSnakeHead» og lim inn denne koden etter at «TSnakeHead» er slutt. Endre navnet til for eksempel «TSnakeTail». Husk å endre sprite-informasjonen lik at du nå viser halen og ikke hodet.

I denne klassen «TSnakeTail» kan du nå fjerne «setDirection» funksjonen. Siden halen til slangen ikke får retning fra brukeren, men henter retningen som hodet hadde før hodet flyttet seg. Så du må nå la hodet fortelle spillet hvilken retning som hodet har. Dette kan du gjøre på flere måter. En løsning kan vær å opprette en tabell som inneholder rader og koloner. Oppdater den spesifikke cellen som hodet har med retning. (Se punkt b i Task 2)

Opprett ett globalt hale-objekt på samme måte som med hode objektet. Og lag sleve forekomsten av klassen i «newGame» også på samme måte som for hodet.

b) Spillebrettet i rader og kolonner:

Siden hodet flytter seg i rader og kolonner er det gunstig å lagre retningen til hodet i denne tabellen: Opprett en global tabell som foreløpig er tom, for eksempel:

```
const gameBoard = [];
```

Lag nå et statisk objekt for eksempel «CellInfo» med disse attributtene:

```
const CellInfo = {direction: EDirection.Right, bait: null, snake: false};
```

I «newGame» funksjonen kan du nå fylle inn alle radene og kolonnene med:

```
Object.create(CellInfo)
```

Bruk det vi har lært om traversering av tabeller med rader og kolonner, bruk en ytre og en indre for-løkke for rader og kolonner.

c) Hode forteller retningen:

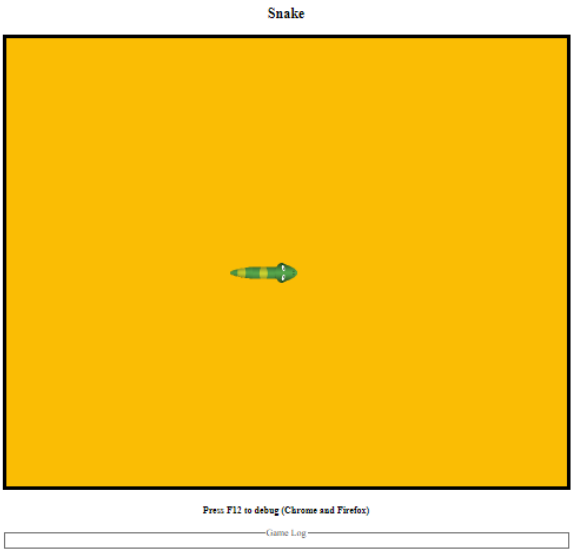
Endre nå «TSnakeHead.move» slik at hodet oppdaterer retningen i celleinformasjonen til spillebrettet så fort funksjonen kjøres. Så halen kan plukke den opp når den kommer til den cellen.

Gjør de nødvendige justeringene i «drawGame» funksjonen slik at du får tegnet ut halen også.

Test programmet og verifiser at halen følger etter hodet og har riktig retning.

TASK 3 — 30 POENG

Vi legger nå inn kroppen til slangen.



a) Kroppen til slangen

Ta en kopi av klassen til halen og lim den inn i koden mellom klassen til hodet og klassen til halen. Kall denne klassen for eksempel for «TSnakeBody».

Kroppen til slangen har 6 unike former som utgjør retningen på kroppen. Fire svinger, en opp/ned og en høyre/venstre. Vi kan bruke disse 6 bildene til å vise alle 10 retningene til slangekroppen. Når slangen svinger så må vi vite hvor den kommer fra slik at vi kan «bøye» kroppen i riktig retning. Lag ett enum-objekt lokalt i toppen av denne klassen som holder på alle 10 retningene og sett nummer på disse i henhold til indeksen som hvert bilde har. Når kroppen flytter seg så må du sjekke på før og etter slik at du kan sette riktig sprite-indeks. Du må lage de nødvendige testene som skal til for å gjøre dette.

b) Forekomst av TSnakeBody

Du kan lage en forekomst på samme måte som for hodet og halen, du vil da ende opp med en kode som tilsvarer dette i «newGame» funksjonen:

```
snakeHead = new TSnakeHead(10, 6);
snakeBody = new TSnakeBody(10, 5)
snakeTail = new TSnakeTail(10, 4);
```

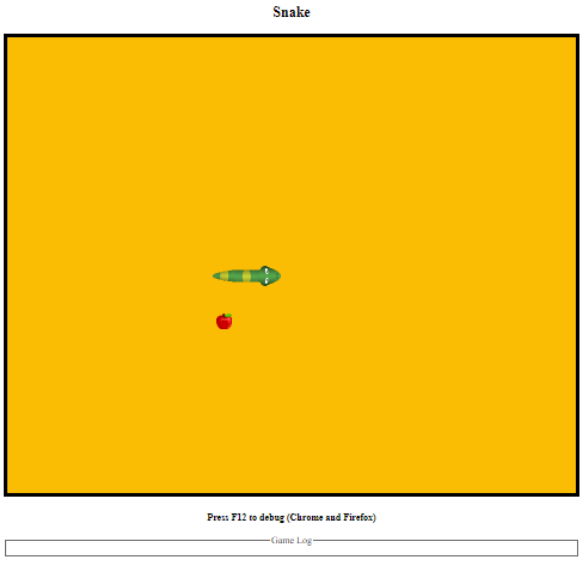
Kjør spillet å se at du har ett hode, en kropp og en hale. Sjekk att alle delenen følger hverandre og at kroppen har riktig retning. Om du ikke klarer å løse kroppens riktige retning men den følger som den skal kan du gå vidre til Task 4, men du vil ikke få full utelling på Task 3.

Her er et hint på hvordan «drawGame» nå kan se ut:

```
function drawGame() {
    snakeHead.move();
    if (gameStatus === EGameStatus.GameOver) {
        window.clearInterval(hndUpdate);
        addLogText("Game Over!");
        return;
    }
    snakeTail.move();
    ctx.clearRect(0, 0, cvs.width, cvs.height);
    snakeHead.draw();
    snakeTail.draw();
}
//-----------------------------------------------
```

TASK 4 — 30 POENG

Vi skal nå legge inn eple i spillet, og få slangen til å vokse når den spiser eplet.



a) Klasse for eplet

Lag en funksjonsbasert klasse og kall den for eksempel for «TBait» denne trenger ingen parametere siden den kan slev plassere seg selv på en ledig plass på brettet.

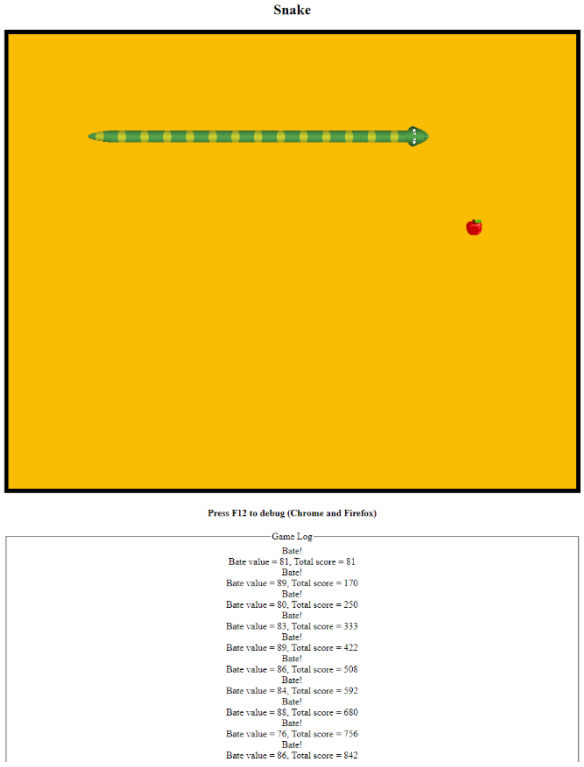
Sett opp de nødvendige medlemmene for å ta i bruk «sprite» på samme måte som for slangen, bare nå med eplet som sprite-informasjon.

Lag nå en «move» funksjon i klassen som setter eplet på en ledig plass på spillebrettet. Du kan løse dette med å lage en «do/while» løkke som tilfeldig plasserer eplet på brettet og så må du kjøre denne løkken så lenge eplet prøver å plassere seg på en opptatt plass, for eksempel om cellen er okkupert av slangen. Du må da ta i bruk tabellen som holder på celleinformasjonen og sette «CelleInfo.snake» til «true» når slange hodet flytter seg, og til «false» etter at halen har flyttet seg.

Når eplet har plassert seg på en ledig plass så kan du nå sette CelleInfo.bait til true, og til false når slangen spiser eplet.

b) La slangen vokse og gi poeng og gå raskere.

Siden eplet styrer poengene til slangen er det greit at denne klassen holder på poengene og øker farten på spillet. La poengvedrdien til eplet minke for hver gang du tegner eplet, men aldri unner o poeng.



Når slangehodet flytter seg til en ny celle så kan du sjekke om det finnes et eplet i denne cellen for eksempel slik: (gameBoard[row][col].bait === true). Da kan du la halen legge till et nytt kroppelement med sin rad og kolonne slik at den nye kroppen havner der som halen står. Men da må du ikke flytte halen ved neste forflyttning. Så kan du flytte eplet med «move» funksjonen til «TBait» Du må også endre litt på hvordan «TSnakeBody» får sin retning i det øyeblikket som halen lager nytt kropp-element. Et hint her; bruk den infromasjonen som ligger i den cellen hvor halen er!

Når eplet blir spist så kan du pårpope en funksjon som du lager global for hele spillet. La denne funksjonen legge til poengene som eplet hadde i det øyeblikket det ble spist til den totale poensummen for spillet. Her er det greit å skrive litt logg-tekst slik at du kan teste dette, før du går videre til neste del om «Spill info og menyenr». For å øke farten til slangen så kan du også gjøre dette i denne funksjonen slik:

```
function updateGameScore(aScore) {
    gameScore += aScore;
    if (gameDelay > 50) {
        gameDelay -= 15;
        window.clearInterval(hndUpdate);
        hndUpdate = window.setInterval(drawGame, gameDelay);
    }
}
//------------------------------------------------------------
```

Test at du får slangen til å vokse når du spiser et eplet, samtidig som slangen øker fart og ople flytterseg til en ny posisjon. Bekreft med eventuelt logg-tekst at poengene øker ettersom epler blir spist.

TASK 5 — 20 POENG

Vi skal nå vise spill informasjon og menyene

Denne delen av oppgaven går utpå å tolke kode. Bruk denne koden som er vist her til å implimentere spill informasjon og menyer.

a) Lage knapper til menyene.

```
function TButton(aPos, aSpriteInfo, aClickFunction) {
    const sp = new TSprite(imgSheet, aSpriteInfo, aPos);        // Sprite object
    const bounds = new TBoundsRectangle(aPos, aSpriteInfo);     // Bounds for mouse hit
    const onClick = aClickFunction;                             // Call back function when button is clicked
    const clickObject = this;                                   // The object that sets the event

    //draw function for updating the button
    this.draw = function () {
        sp.draw();
    };

    //Mouse over function
    this.mouseOver = function (aMousePos) {
        return bounds.hitPos(aMousePos);
    };

    //Mouse click function
    this.click = function (aMousePos) {
        if (bounds.hitPos(aMousePos) && onClick) {
            onClick(clickObject);
        }
    }
}
```

Du oppretter forekomster av klassen slik:

```
const center = new TPosition(cvs.width / 2, cvs.height / 2);
let pos = new TPosition(center.x - (SnakeSheet.Play.w / 2), center.y - (SnakeSheet.Play.h / 2));
btnPlay = new TButton(pos, SnakeSheet.Play, newGame);
```

Og bruker muse-eventene slik:

```
function cvsMouseMove(aEvent) {
    // Mouse move over canvas
    setMousePos(aEvent);
    switch (gameStatus) {
        case EGameStatus.New:
            if (btnPlay.mouseOver(mousePos)) {
                cvs.style.cursor = "pointer";
            } else {
                cvs.style.cursor = "auto";
            }
            break;
        case EGameStatus.GameOver:
            if (btnHome.mouseOver(mousePos)) {
                cvs.style.cursor = "pointer";
            } else if (btnRetry.mouseOver(mousePos)) {
                cvs.style.cursor = "pointer";
            } else {
                cvs.style.cursor = "auto";
            }
            break;
    }
}
//---------------------------------------------------------------------

function cvsClick(aEvent) {
    switch (gameStatus) {
        case EGameStatus.New:
            btnPlay.click(mousePos);
            break;
        case EGameStatus.GameOver:
            btnHome.click(mousePos);
            btnRetry.click(mousePos);
            break;
        case EGameStatus.Pause:
            btnResume.click(mousePos);
            break;
    }
    cvs.style.cursor = "auto";
}
//---------------------------------------------------------------------
```

b) Vise spill-poengene og verdien av eplet:

Implementering av klassen:

```
function TGameInfo(aPos, aNumber) {
    const pos = aPos;
    const spl = [];
    let txtNumber = "000000";
    const scale = {x: 0.5, y: 0.5};
    for (let i = 0; i < txtNumber.length; i++) {
        const element = {pos: null, sp: null};
        element.pos = new TPosition((pos.x + 3 + (i * SnakeSheet.Number.w)), pos.y);
        element.sp = new TSprite(imgSheet, SnakeSheet.Number, element.pos);
        element.sp.setScale(scale);
        spl.push(element);
    }

    this.draw = function () {
        let alpha;
        if (gameStatus === EGameStatus.GameOver) {
            alpha = 1;
            scale.x = 1;
            scale.y = 1;
        } else {
            alpha = 0.3
            scale.x = 0.5;
            scale.y = 0.5;
        }
        for (let i = 0; i < spl.length; i++) {
            spl[i].pos.x = pos.x + 3 + (i * SnakeSheet.Number.w * scale.x);
            spl[i].pos.y = pos.y * scale.y;
            const index = parseInt(txtNumber.charAt(i));
            spl[i].sp.setIndex(index);
            spl[i].sp.setAlpha(alpha);
            spl[i].sp.setScale(scale);
            spl[i].sp.draw();
        }
    };

    this.setNumber = function (aNumber) {
        txtNumber = aNumber.toString();     // 5 --> 005
        this.draw();
    };

    this.setPos = function (aX, aY) {
        pos.x = aX;
        pos.y = aY;
        this.draw();
    };

    this.setNumber(aNumber);

} // End of TGameInfo Class
```

Forekomst av klassen:

```
baitScoreInfo = new TGameInfo(new TPosition(6, 10), 0);
scoreInfo = new TGameInfo(new TPosition(5, 100), 0);
```

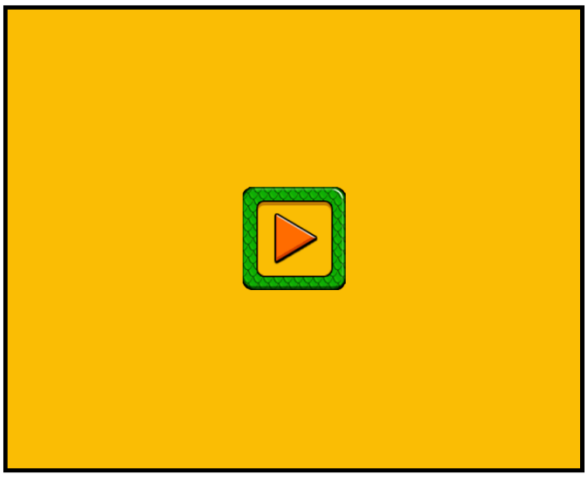
Bruken av forekomstene:

```
scoreInfo.setNumber(gameScore);
```

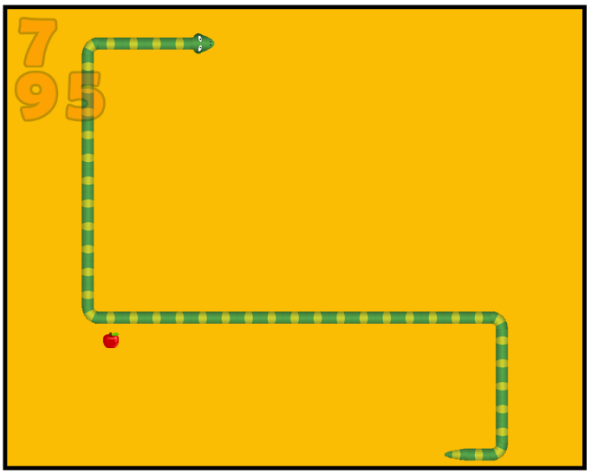
og

```
scoreInfo.setPos(5, 100);
```

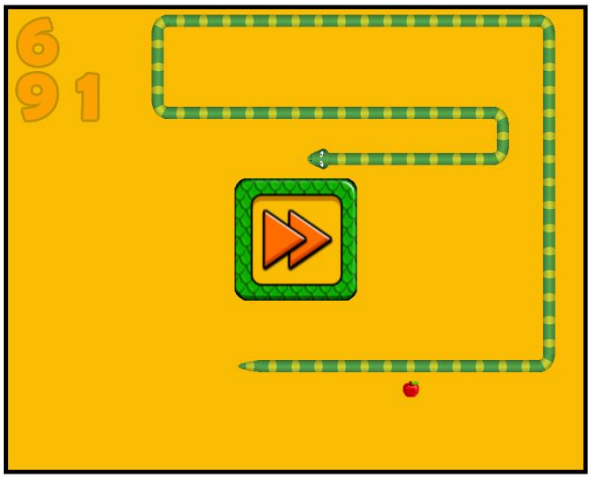
ILLUSTRASJONER



Figur 1: Føre spille starter.



Figur 2: Mens spillet pågår



Figur 3: Spillet er pauset

Figur 4: Spillet er slutt