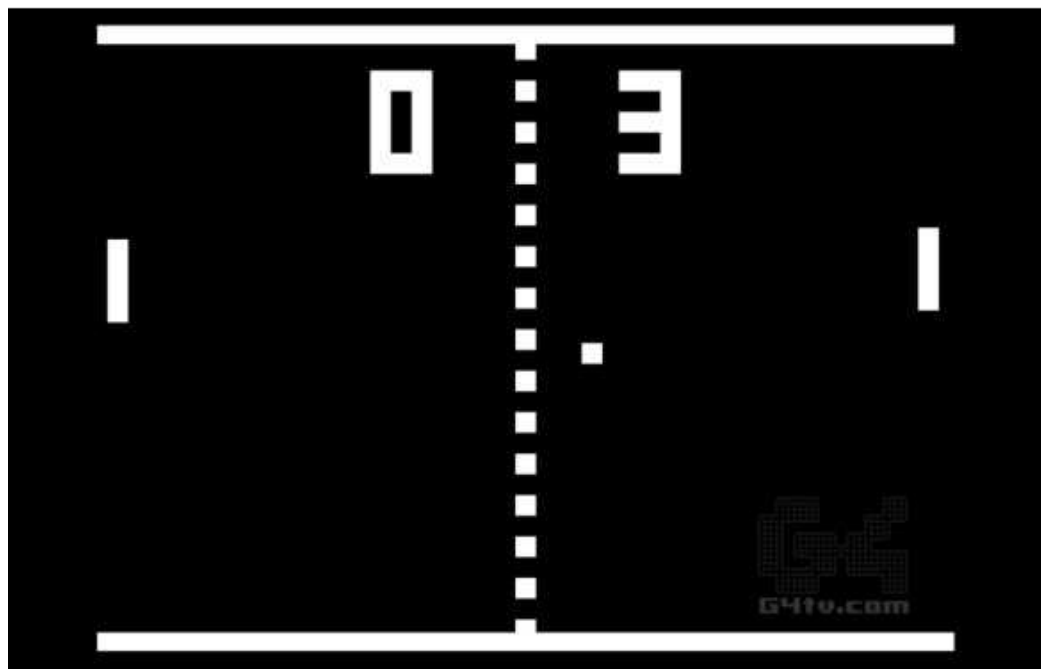


Programmer un pong



Pourquoi un pong ?



- Un des premiers jeux vidéos
- Concepts de base de la plupart des jeux qui ont suivi
- Un jeu d'interactions en temps réel

Rappel : la BASE d'un jeu vidéo

```
fenêtre {  
    load();  
  
    boucle {  
        update();  
        draw();  
    };  
}
```

COMPLETED

La balle

Dessiner une balle

- main.js draw()

```
function draw() {  
  ...  
  // Draw game  
  // Ball  
  canvasContext.fillStyle = 'white';  
  canvasContext.beginPath();  
  canvasContext.arc(50, 50, 10, 0, Math.PI * 2, true);  
  canvasContext.fill();  
}
```

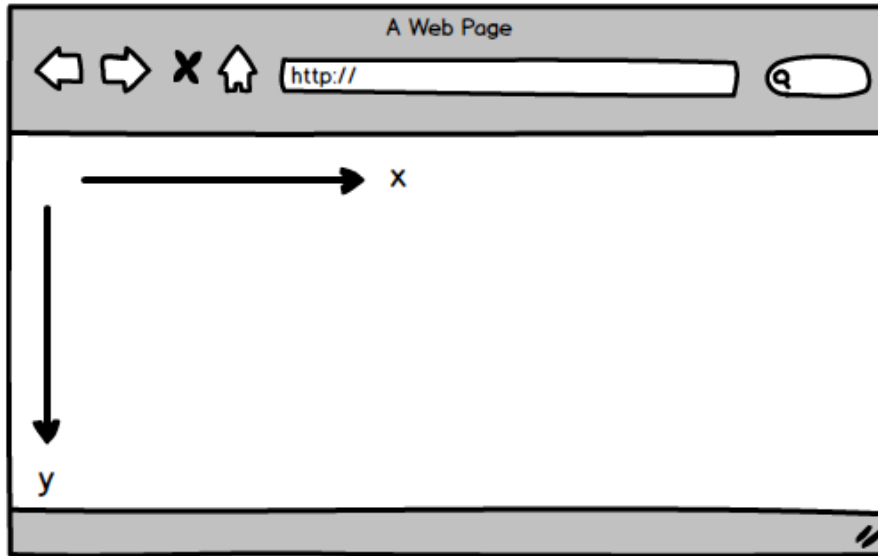
file:///D:/Sync/Prototypes/Cours/02 - Pong/code/step01_display-ball/index.html

Most Visited Getting Started Devlogs - itch.io



Faire bouger la balle : design

- Variables pour stocker les coordonnées de la balle.



- Donner une valeur initiale à ces variables
- Changer le code pour dessiner la balle à ces coordonnées
- Faire bouger la balle avec le code suivant :

```
ballX = ballX + 1;  
ballY = ballY + 1;
```

Faire bouger la balle : code

- main.js

```
...  
let ballX, ballY;  
  
function load() {  
  ...  
  // Loading  
  ballX = 50;  
  ballY = 50;  
}
```

- main.js update()

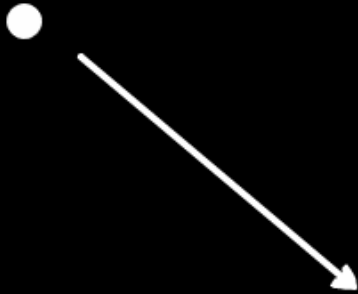
```
function update() {  
  // Move ball  
  ballX = ballX + 1;  
  ballY = ballY + 1;  
}
```

- main.js draw()

```
function draw() {  
  ...  
  canvasContext.arc(ballX, ballY, 10, 0, Math.PI * 2, true);  
  canvasContext.fill();  
}
```


file:///D:/Sync/Prototypes/Cours/02 - Pong/code/step01_display-ball/index.html

Most Visited Getting Started Devlogs - itch.io



Vitesse variable de la balle : design

- Plutôt que d'avoir une vitesse fixe, on veut une vitesse variable
- Cela nous servira notamment à faire accélérer ou rebondir la balle
- On veut une vitesse verticale et une vitesse horizontale
- Ne pas oublier d'initialiser les vitesses
- ... et de les utiliser dans le code

Vitesse variable de la balle : code

- main.js

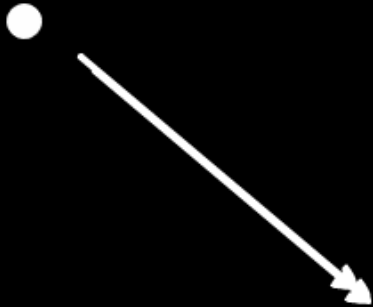
```
...  
let ballX, ballY;  
let speedX, speedY;  
  
function load() {  
...  
    // Loading  
    ballX = 50;  
    ballY = 50;  
    speedX = 2;  
    speedY = 2;  
}
```

- main.js update()

```
function update() {  
    // Move ball  
    ballX = ballX + speedX;  
    ballY = ballY + speedY;  
}
```

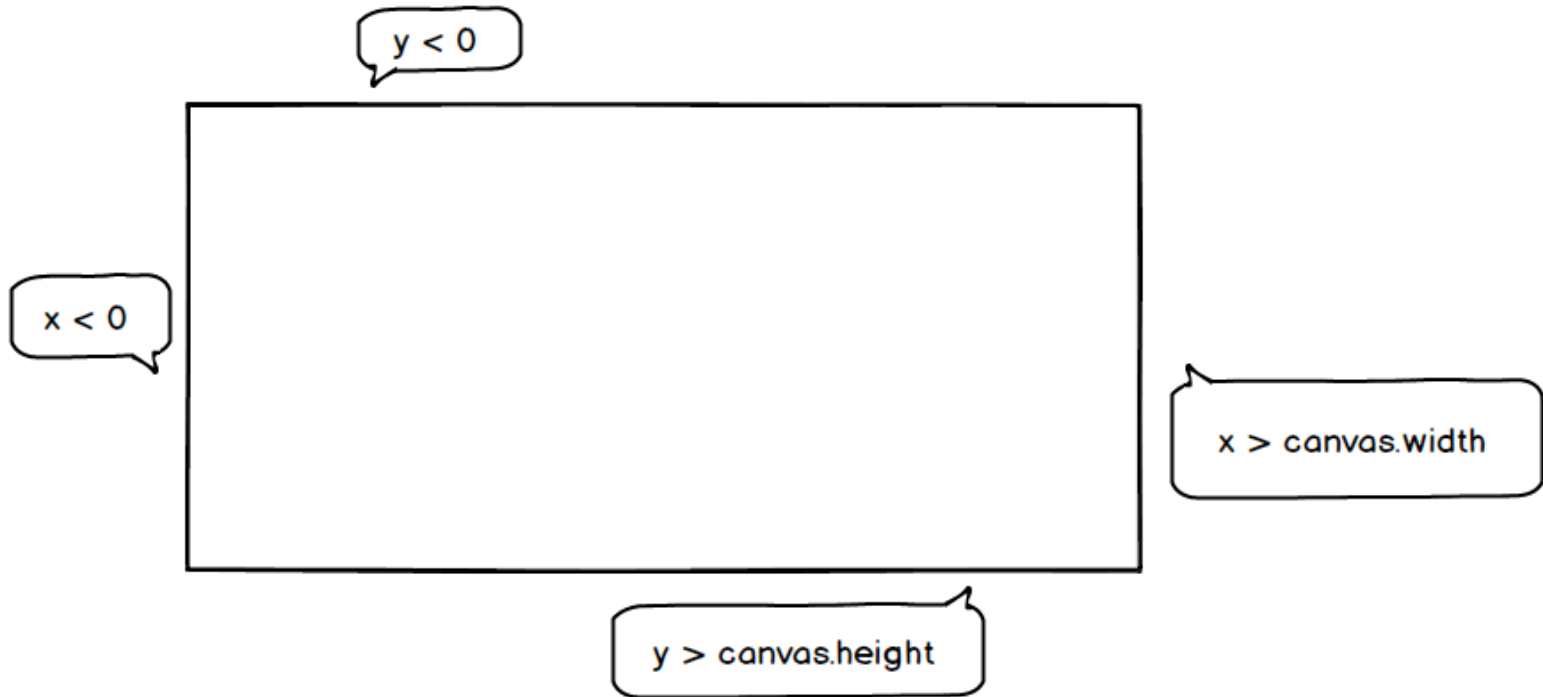
file:///D:/Sync/Prototypes/Cours/02 - Pong/code/step01_display-ball/index.html

Most Visited Getting Started Devlogs - itch.io



Rebond : design

- La balle rebondit si elle atteint un bord de l'écran
- Les bords de l'écran sont atteints quand on dépasse certaines coordonnées



- Pour tester si on dépasse une coordonnée, on utilise `if(...)`. Par exemple :

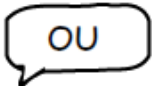
```
if (ballX > canvas.width) {  
    // comportement  
}
```

- Pour faire rebondir une balle, on inverse sa vitesse (horizontale ou verticale). Inversion d'une variable :

```
variable = -variable;
```

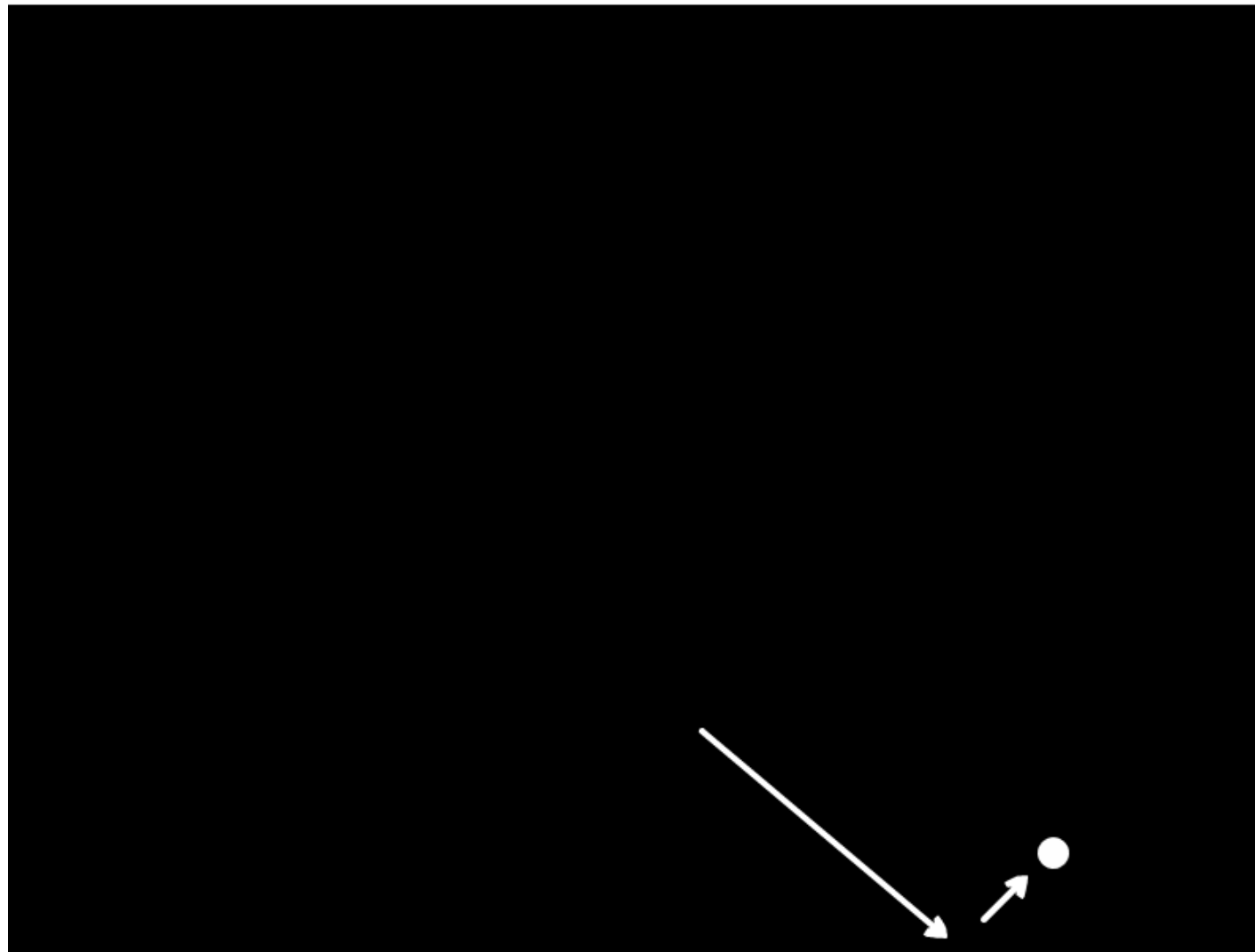
Rebond : code

- main.js update()

```
function update() {  
  ...  
  // Bounce   
  if (ballX > canvas.width || ballX < 0) {  
    speedX = -speedX;  
  }  
  if (ballY > canvas.height || ballY < 0) {  
    speedY = -speedY;  
  }  
}
```

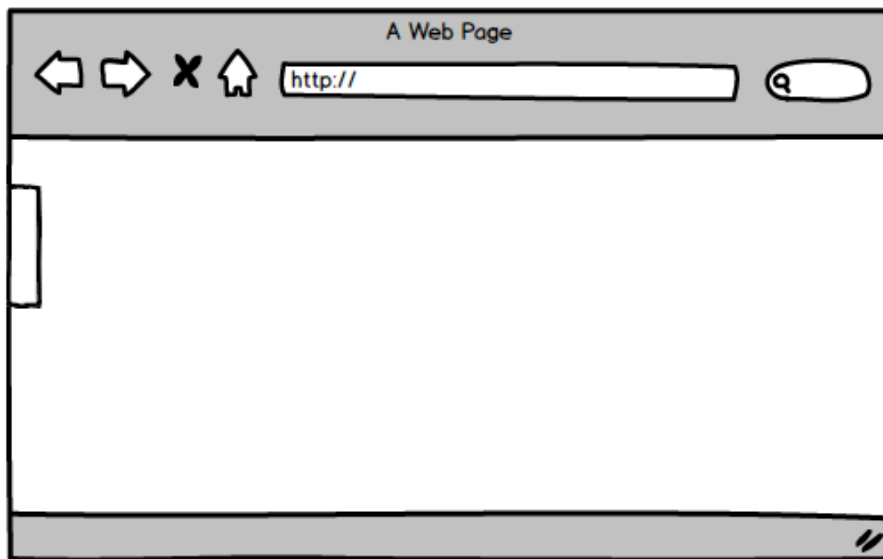
file:///D:/Sync/Prototypes/Cours/02 - Pong/code/step04_bouncing-ball/index.html

Most Visited Getting Started Devlogs - itch.io



Dessiner le paddle : design

- Variables pour stocker les coordonnées de la balle.



- Donner une valeur initiale à ces variables. La coordonnées x du paddle est égale à 0.
- Dessiner le paddle à ses coordonnées avec le code suivant. (Le paddle fait 20 de large et 100 de haut.)

```
canvasContext.beginPath();  
canvasContext.rect(paddleX, paddleY, 20, 100);  
canvasContext.fill();
```


Dessiner le paddle : code

- main.js Déclarer et initialiser les variables :

```
...  
let paddleX, paddleY;  
...  
function load() {  
...  
    paddleX = 0;  
    paddleY = 100;  
}
```

- main.js draw()

```
function draw() {  
...  
    // Paddle  
    canvasContext.beginPath();  
    canvasContext.rect(paddleX, paddleY, 20, 100);  
    canvasContext.fill();  
}
```

file:///D:/Sync/Prototypes/Cours/02 - Pong/code/step05_paddle/index.html

Most Visited Getting Started Devlogs - itch.io



Déplacer le paddle : design

- On a seulement besoin de vitesse verticale (paddleSpeedY).
- Il nous faut deux variables keyUp et keyDown.
- Il faut déplacer le paddle vers le haut si keyUp est vraie, et vers le bas si keyDown est vraie.
- Les codes pour les flèches haut et bas sont respectivement 38 et 40.
- On veut que la vitesse du paddle soit modifiée si on appuie sur la flèche du haut ou du bas, et qu'elle soit remise à 0 sinon.
- Utiliser pour cela les conditions :

```
if ( /* condition pour se deplacer vers le haut */ ) {  
  
} else if ( /* condition pour se deplacer vers le bas */ ) {  
  
} else {  
  
}
```

Déplacer le paddle : code

- main.js Déclarer et initialiser les variables :

```
...  
let paddleX, paddleY;  
...  
function load() {  
...  
    paddleX = 0;  
    paddleY = 100;  
}
```

- main.js draw()

```
function draw() {  
...  
    // Paddle  
    canvasContext.beginPath();  
    canvasContext.rect(paddleX, paddleY, 20, 100);  
    canvasContext.fill();  
}
```

file:///D:/Sync/Prototypes/Cours/02 - Pong/code/step05_paddle/index.html

Most Visited Getting Started Devlogs - itch.io





Bug : le paddle ne s'arrête pas



- Explication

```
function keyDetect(e) {  
  if (e.keyCode == 38) {  
    keyUp = true;  
  }  
  if (e.keyCode == 40) {  
    keyDown = true;  
  }  
}
```

```
function update() {  
  ...  
  if (keyUp) {  
    paddleSpeedY = -10;  
  } else if (keyDown) {  
    paddleSpeedY = 10;  
  } else {  
    paddleSpeedY = 0;  
  }  
}
```



Jamais exécuté

Améliorer les contrôles : design

- Le problème est que la variable `pressedKey` reste la même quand on relache la touche de déplacement
- On va donc détecter l'évènement "relacher une touche" et lui faire changer la valeur de `pressedKey`
- Pour détecter un l'évènement et lancer la fonction `releaseKey` :

```
document.addEventListener('keyup', releaseKey, false);
```

- Créer une fonction `releaseKey` dont le prototype sera :

```
function releaseKey(e) {  
  
}
```

Améliorer les contrôles : code

- main.js load()

```
function load() {  
  ...  
    document.addEventListener('keyup', releaseKey, false);  
  ...  
}
```

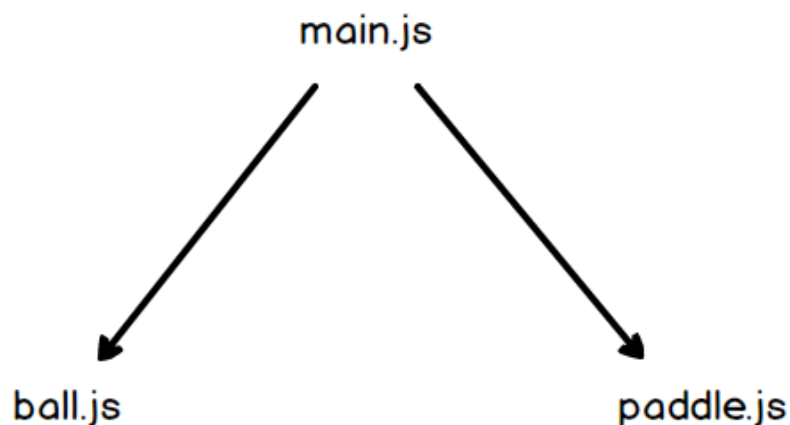
- main.js

```
function releaseKey(e) {  
  if (e.keyCode == 38) {  
    keyUp = false;  
  }  
  if (e.keyCode == 40) {  
    keyDown = false;  
  }  
}
```

- Il restera des bugs, mais le jeu sera globalement jouable. Construire une véritable gestion des contrôle sortirait du cadre de ce cours.

Ca commence à être le bazar !

- On a beaucoup de variables
- Imaginons qu'on veuille créer un deuxième pad, il va falloir copier/coller, risquer de se tromper etc.
- Solution :



Extraire le code de la balle et du paddle pour les mettre dans un fichier différent, et en faire des OBJETS !

Classes : Ball (1/4)

- Une classe est morceau de code informatique qui enferme un concept et tous ses éléments
- Par exemple pour la balle : ses coordonnées, sa vitesse, son comportement, le fait d'être dessinée...
- Le prototype de la classe Ball dans un nouveau fichier ball.js :

```
class Ball {  
  
  constructor(x, y, radius, speedX, speedY) {  
    this.x = x;  
    this.y = y;  
    this.radius = radius;  
    this.speedX = speedX;  
    this.speedY = speedY;  
  }  
  
  update() {  
  
  }  
  
  draw() {  
  
  }  
}
```

Classes : Ball (2/4)

- Il faut maintenant copier et adapter le contenu de update et draw depuis le main.js
- Utiliser ctrl + H pour rechercher ballX et remplacer par this.x

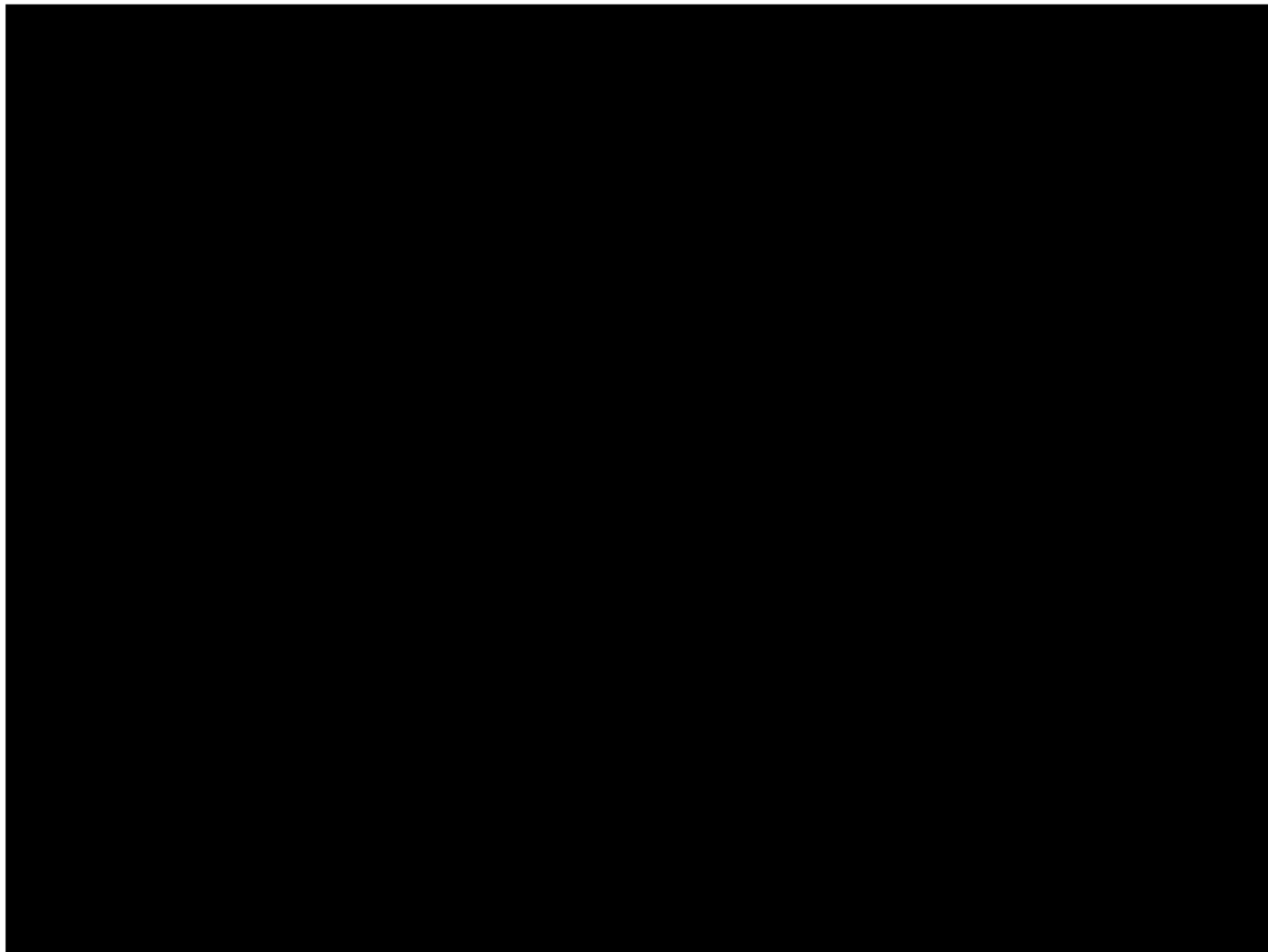
```
class Ball {  
...  
  update() {  
    this.x = this.x + this.speedX;  
    this.y = this.y + this.speedY;  
  
    if (this.x > canvas.width || this.x < 0) {  
      this.speedX = -this.speedX;  
    }  
    if (this.y > canvas.height || this.y < 0) {  
      this.speedY = -this.speedY;  
    }  
  }  
  
  draw() {  
    canvasContext.fillStyle = 'white';  
    canvasContext.beginPath();  
    canvasContext.arc(this.x, this.y, this.radius, 0, Math.PI * 2, true);  
    canvasContext.fill();  
  }  
}
```

Classes : Ball (3/4)

- Il faut maintenant appeler le code de la classe depuis main.js

```
let ball;  
...  
  
function load() {  
...  
    // Loading  
    ball = new Ball(50, 50, 10, 2, 2);  
...  
}  
  
function update() {  
    ball.update();  
...  
}  
  
function draw() {  
...  
    // Draw game  
    ball.draw();  
...  
}
```

← ⓘ | file:///home/gaetz/Sync/Prototypes/Cours/02 - Pong/code/step08_ball-class/index.html



Inspector Console Débugueur {} Éditeur de style Performances Mémoire Réseau

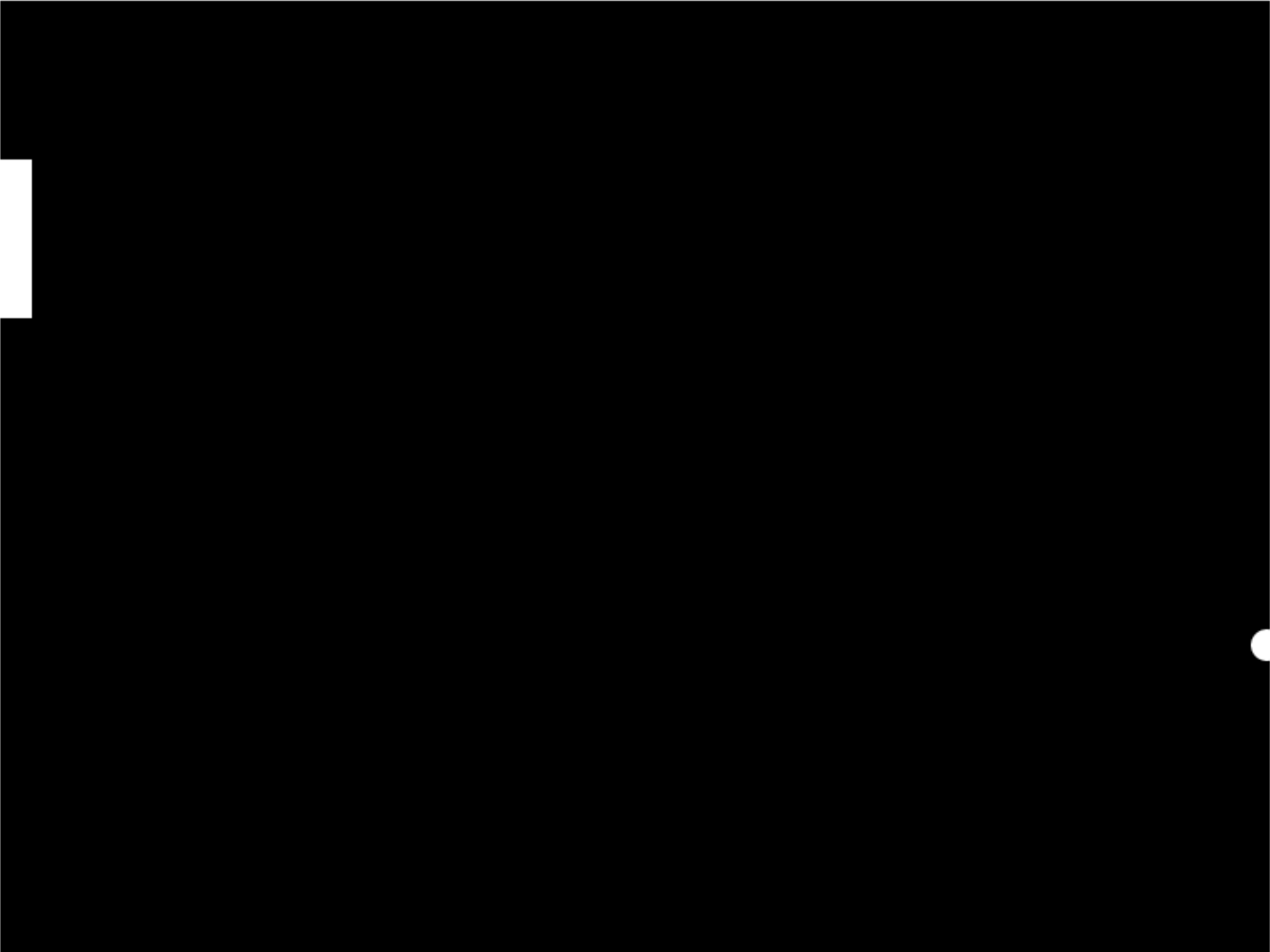
🗑 Réseau CSS JS Sécurité Journal Serveur

✖ ▶ ReferenceError: Ball is not defined [\[En savoir plus\]](#)

Classes : Ball (4/4)

- On a pas appelé le fichier ball.js depuis le index.html :)

```
...  
<body>  
  <script src="main.js"></script>  
  <script src="ball.js"></script>  
  <canvas id="gameCanvas" width="800" height="600"  
style="background: black;"></canvas>  
</body>  
...
```



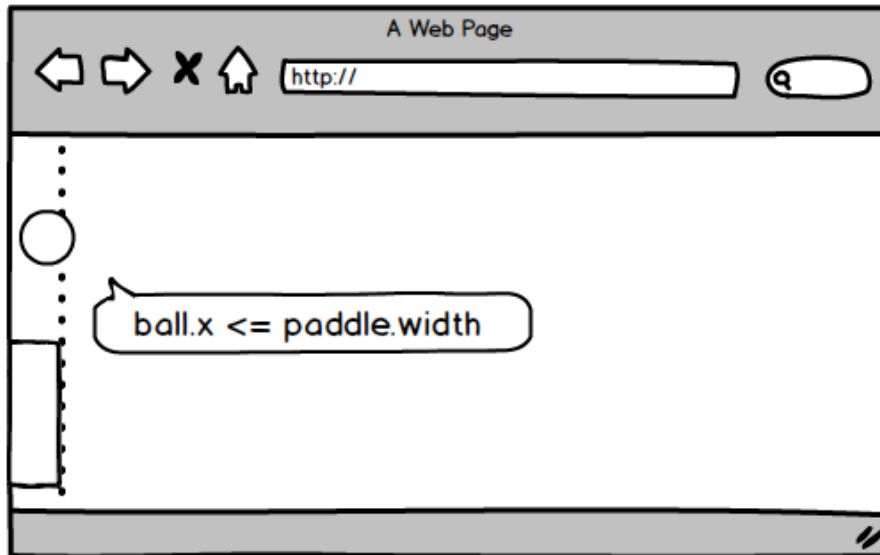
Classes : Paddle

- Faire la même chose pour le paddle
- Voici le prototype :

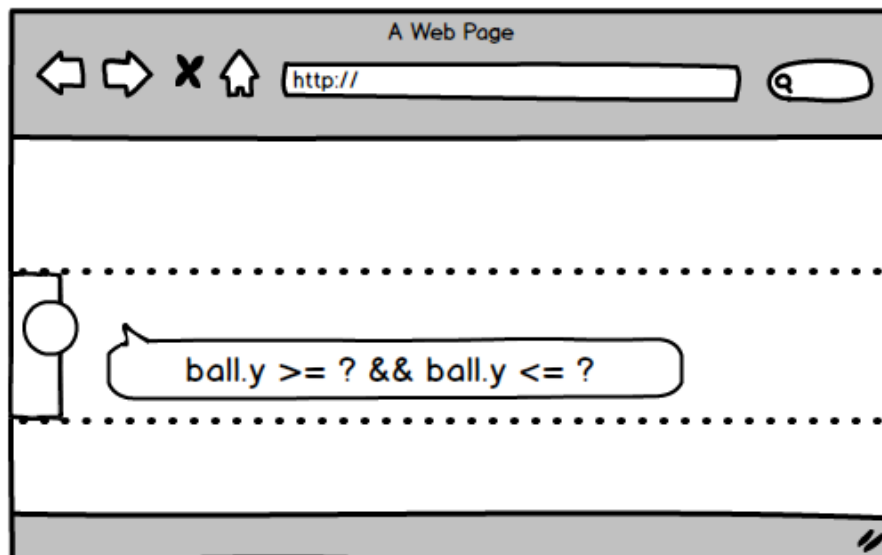
```
class Paddle {  
  
    constructor(x, y, width = 20, height = 100, speedY = 0) {  
        this.x = x;  
        this.y = y;  
        this.width = width;  
        this.height = height;  
        this.speedY = speedY;  
    }  
  
    update() {  
  
    }  
  
    draw() {  
  
    }  
}
```


Faire rebondir la balle sur le paddle : design

- Supprimer le rebond sur le bord gauche de l'écran
- On teste si la balle est à portée du paddle



- Si c'est le cas, on teste si la balle est "dans le paddle"



- Si c'est le cas, on fait rebondir la balle

Faire rebondir la balle sur le paddle : code

- ball.js, créer une méthode padBounce()

```
function padBounce() {  
  this.speedX = -this.speedX  
}
```

- ball.js, supprimer le rebond sur le bord gauche

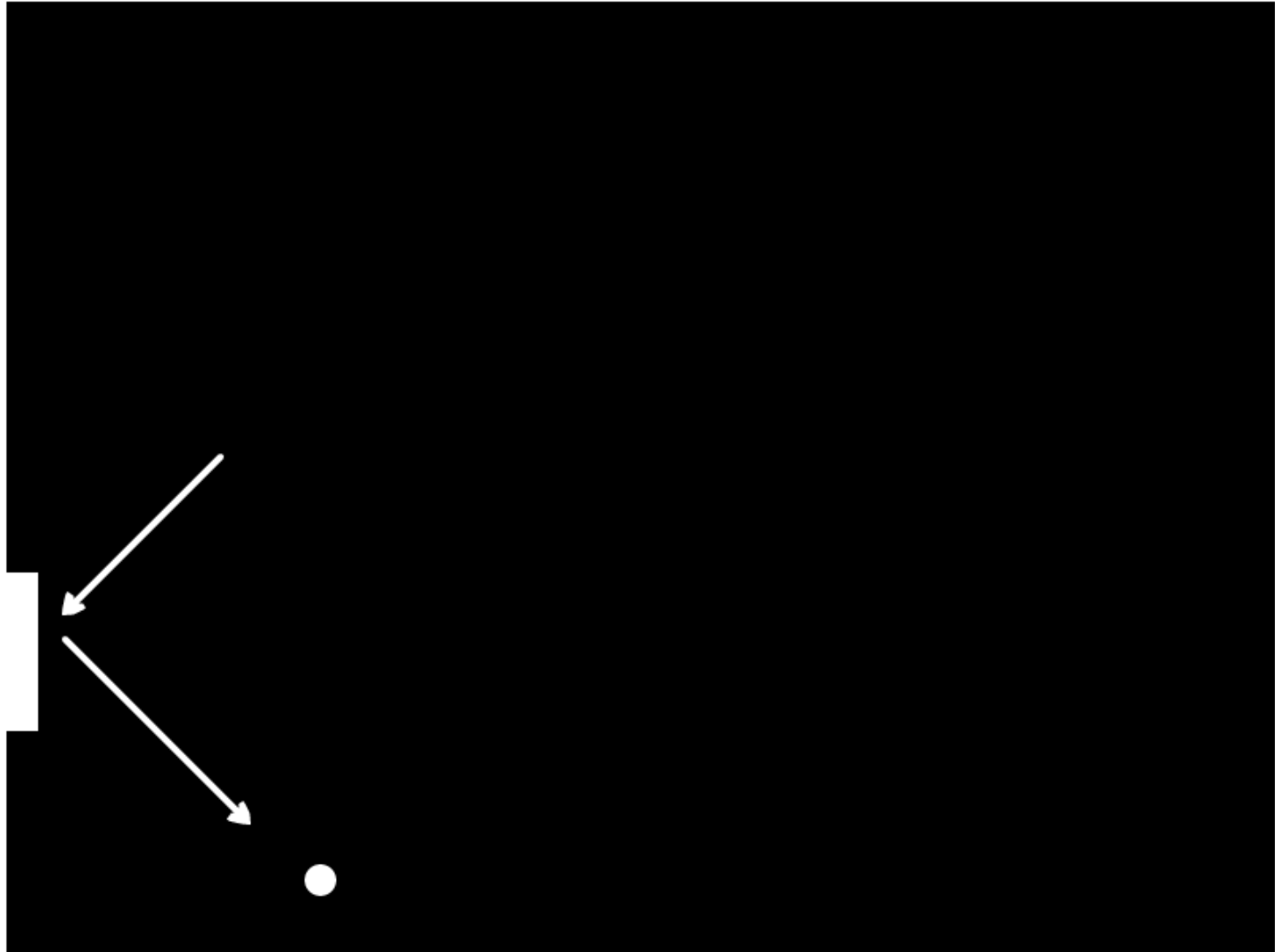
```
function update() [  
  ...  
    if (this.x > canvas.width) {  
      this.speedX = -this.speedX;  
    }  
  ...  
]
```



On ne teste
plus $x < 0$

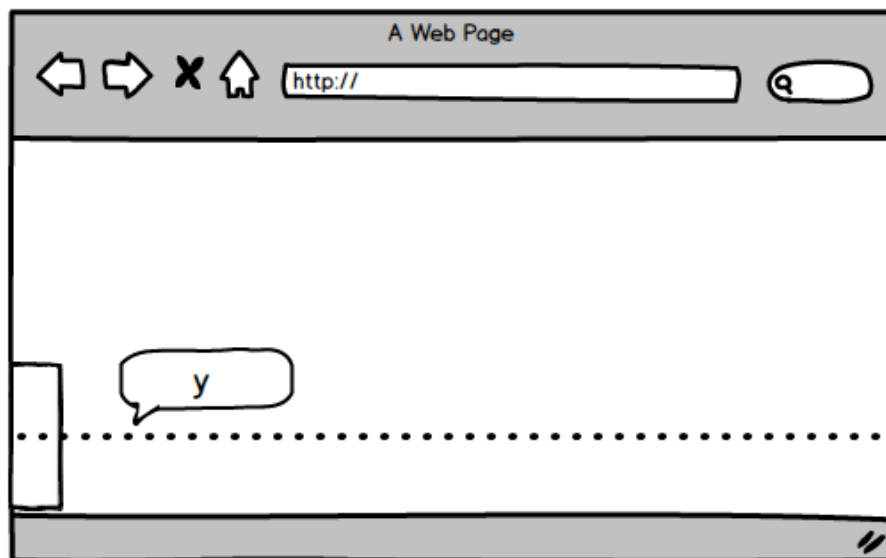
- main.js, tester les conditions de rebond

```
function update() {  
  ...  
  // Pad bounce  
  if(ball.x <= paddle.width) {  
    if(ball.y >= paddle.y && ball.y <= paddle.y + paddle.height) {  
      ball.padBounce();  
    }  
  }  
}
```



Limiter le déplacement du paddle : design

- On veut empêcher le paddle de sortir de l'écran vers le haut ou vers le bas
- Il faut empêcher le paddle de répondre aux commandes si ses coordonnées ont atteint les limites
- Prendre en compte la hauteur du paddle

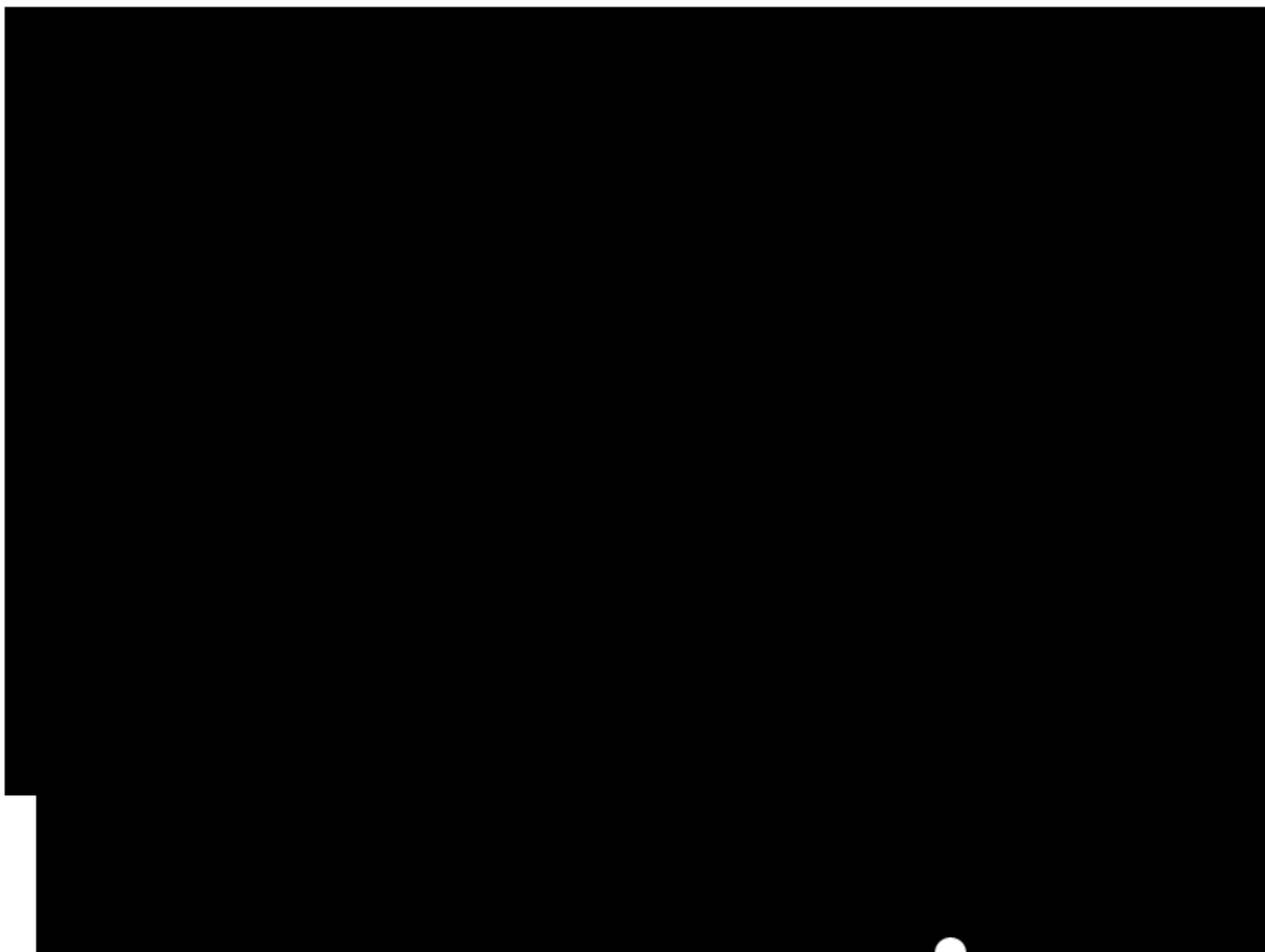


- Prévoir le cas où cette limite serait quand même dépassée

Limiter le déplacement du paddle : code

- paddle.js

```
function update() {  
  ...  
  // Move  
  if(this.y >= 0 && this.y <= canvas.height - this.height) {  
    this.y = this.y + this.speedY;  
  }  
  if(this.y < 0) {  
    this.y = 0;  
  }  
  if(this.y > canvas.height - this.height) {  
    this.y = canvas.height - this.height;  
  }  
}
```

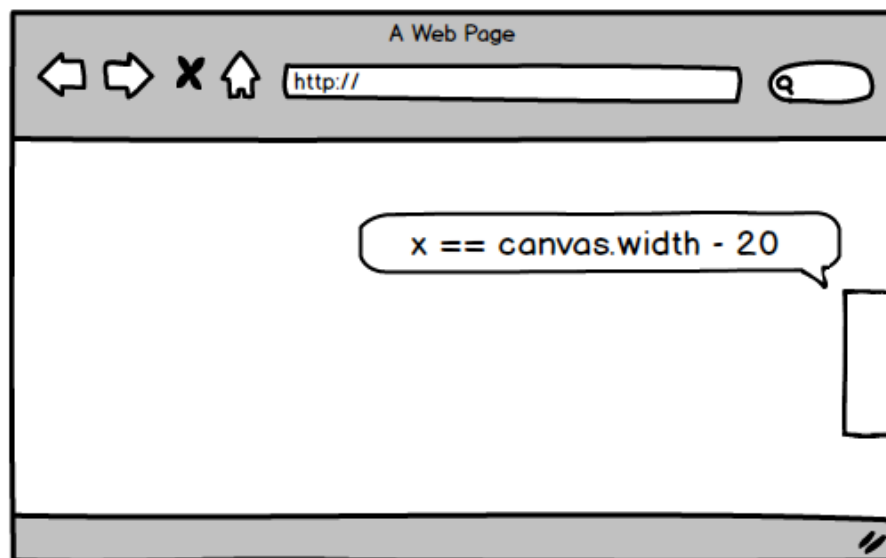


Creation du paddle IA : design

- On veut créer un paddle contrôlé par l'ordinateur. Pour l'instant, on souhaite juste qu'il se déplace pour rester en face de la balle.
- On va étendre la classe Paddle pour la réutiliser, et simplement changer le contenu de la méthode update

```
class PaddleAI extends Paddle {  
  update() {  
    ...  
  }  
}
```

- Le paddle sera placé ainsi :



- Ne pas oublier d'incluer le fichier paddleAI.js créé dans le fichier html, de construire le PaddleAI, d'appeler update() et draw()

Creation du paddle IA : code

- paddleAI.js

```
class PaddleAI extends Paddle {  
  
  update() {  
    if(ball.y >= this.y + this.height) {  
      this.speedY = 5;  
    } else if (ball.y <= this.y) {  
      this.speedY = -5;  
    } else {  
      this.speedY = 0;  
    }  
    this.y = this.y + this.speedY;  
  }  
}
```

- main.js

```
let paddleAI;  
...  
paddleAI = new PaddleAI(canvas.width - 20, 100);  
...  
paddleAI.update();  
...  
paddleAI.draw();  
...
```

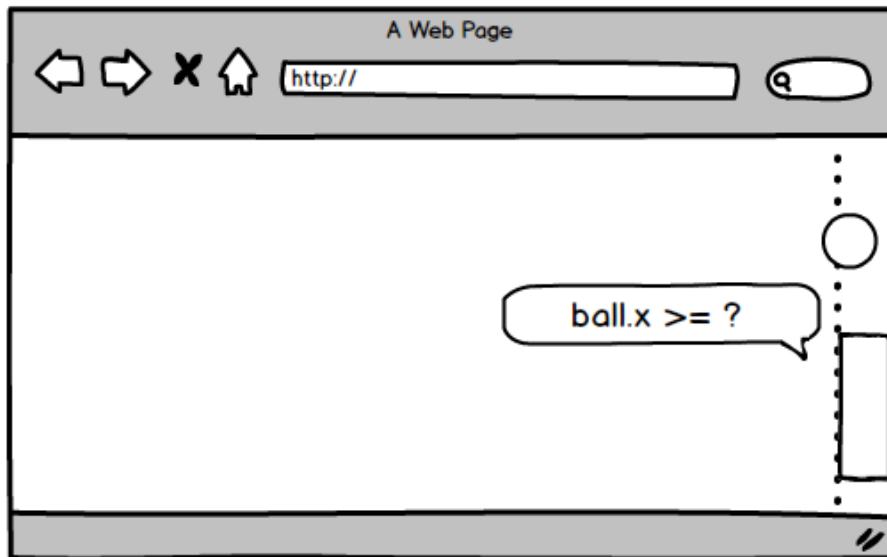
The diagram shows three horizontal arrows pointing from the code to the right:

- An arrow from `paddleAI = new PaddleAI(canvas.width - 20, 100);` points to the text `dans load()`.
- An arrow from `paddleAI.update();` points to the text `dans update()`.
- An arrow from `paddleAI.draw();` points to the text `dans draw()`.



Faire rebondir la balle sur le paddle IA : design

- Supprimer le rebond de la balle sur le bord droite de l'écran
- Comme pour le paddle du joueur, on teste si la balle est à portée du paddle

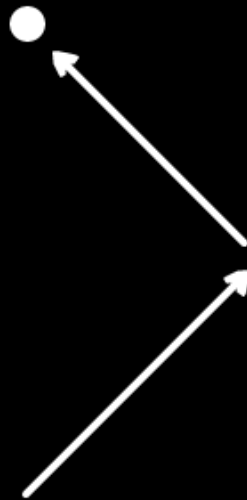


- La suite est similaire.

Creation du paddle IA : code

- main.js update()

```
...  
    if(ball.x >= canvas.width - paddleAI.width) {  
        if(ball.y >= paddleAI.y && ball.y <= paddleAI.y + paddleAI.height) {  
            ball.padBounce();  
        }  
    }  
}
```



Terminer la boucle de gameplay : design

- Plus qu'une étape pour arriver à un Pong simpliste mais fonctionnel : faire réapparaître la balle
- Il s'agit simplement de tester si la balle sort de l'écran, et de la remettre en place si besoin
- Utiliser une fonction, on pourra améliorer cette fonctionnalité plus tard (scoring...)

Terminer la boucle de gameplay : code

- main.js :

```
function update() {  
  ...  
  if(ball.isOutOfScreen()) {  
    respawn();  
  }  
}  
  
function respawn() {  
  ball.x = canvas.width / 2;  
  ball.y = canvas.height / 2;  
  ball.speedX = -ball.speedX;  
}
```

- ball.js

```
isOutOfScreen() {  
  return this.x < 0 || this.x > canvas.width;  
}
```



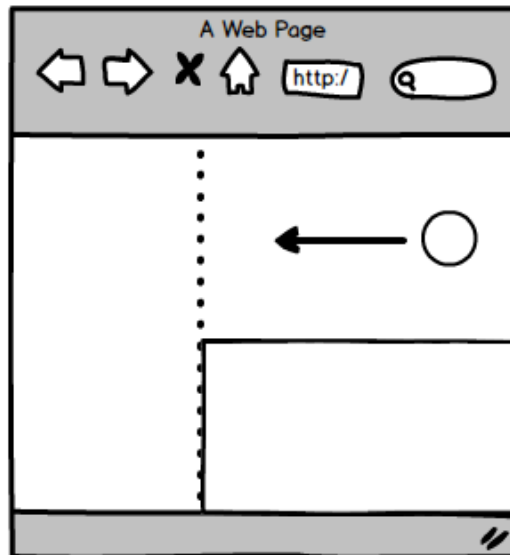
Bug : la balle peut rester bloquée



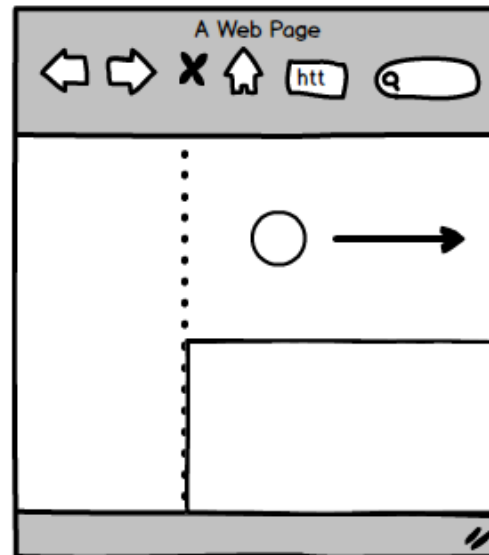
- Apparition du bug quand j'ai mis les vitesses de la balle à 5



- Que se passe t-il ? La balle a dépassé la largeur du paddle, inverse sa vitesse horizontale, mais cela ne lui suffit pas pour sortir de la zone de largeur du paddle, et elle inverse à nouveau sa vitesse. Allers-retours.



`ball.x >= canvas.width - paddleAI.width`



`ball.x >= canvas.width - paddleAI.width`

La balle peut rester bloquée : solution

- ball.js

```
padBounce() {  
    this.speedX = -this.speedX;  
    // Bug fix  
    if(this.x > canvas.width - paddleAI.width) {  
        this.x = canvas.width - paddleAI.width;  
    }  
    if(this.x < paddle.width) {  
        this.x = paddle.width;  
    }  
}
```


Le delta-time : rendre le jeu jouable quand il rame

- Simuler un jeu qui rame en remplaçant 1000/60 par 1000/10 (10fps) dans main.js :

```
window.onload = function () {  
  load();  
  setInterval(() => {  
    update();  
    draw();  
  }, 1000 / 10);  
}
```

- On voit que tout le jeu est ralenti
- Solution : calculer le temps entre chaque frame et multiplier tous les déplacements par cette durée
- Cette durée est appelée "delta time", une variable souvent nommée delta ou dt dans les moteurs de jeu

Delta time : implémentation (1/2)

- main.js

```
window.onload = function () {  
  let lastUpdate = Date.now();  
  let now, dt;  
  load();  
  setInterval(() => {  
    now = Date.now();  
    dt = (now - lastUpdate) * 1 / (1000 / 60);  
    lastUpdate = now;  
  
    update(dt);  
    draw();  
  }, 1000 / 60);  
}
```

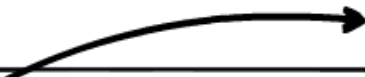
→ durée théorique d'une frame

Delta time : implémentation (2/2)

• main.js

```
function update(dt) {  
    ball.update(dt);  
    paddle.update(dt);  
    paddleAI.update(dt);  
    ...  
}
```

Passage d'un argument (dt n'est pas global)



• ball.js

```
update(dt) {  
    // Move ball  
    this.x = this.x + this.speedX * dt;  
    this.y = this.y + this.speedY * dt;  
    // Bounce  
    ...  
}
```

• paddle.js

```
update(dt) {  
    ...  
    // Move  
    if(this.y >= 0 && this.y <= canvas.height - this.height) {  
        this.y = this.y + this.speedY * dt;  
    }  
    ...  
}
```

• paddleAI.js

```
update(dt) {  
    ...  
    this.y = this.y + this.speedY * dt;  
}
```



Problème : le jeu peut durer indéfiniment



- On va mettre de nouvelles fonctionnalités en place :
 1. Faire rebondir la balle avec un angle différent en fonction de l'endroit où on tape sur le paddle
 2. Augmenter la vitesse de la balle à chaque fois qu'elle rebondit
 3. Ajouter un système de score et terminer le jeu en trois points
 4. Faire apparaître la balle dans une position aléatoire (mais correcte pour les joueurs)