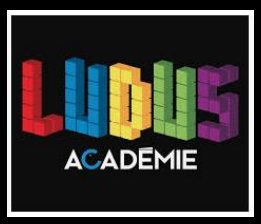


Lab0

Document de Cadrage

Problématique

Comment concevoir un fan-game en utilisant un nouveau framework ?

Conçu par Prob Bastien, le 24/11/2021

Projet Web – Mastère, années 2021/2022

Sommaire

Définition du projet

Pitch :

La ville de Zaun est hors de contrôle ! Les forces militaires de la ville voisine, Piltover, ont dépêché sur les lieux une machine de guerre aussi ingénieuse que chirurgicale. Frayez-vous un chemin dans les entrailles de la ville industrielle et remettez de l'ordre dans ce chaos.

Lab0 se veut être un jeu mélangeant des **mécaniques simples** et **fun** dans un univers déjà conçu à partir de la série mondialement connue *Arcane*. Il n'y aura qu'une partie de jeu unique, où le joueur évoluera dans un décor dystopique mêlant chimie nauséabonde et industrie froide. La **dualité**, thème souligné dans la série, sera reprise en ne proposant au joueur que 2 types d'armes et 2 types d'obstacles.

But du jeu :

Le joueur doit manœuvrer une tourelle, conçue à partir d'engrenages, de tubes chimiques et de cristaux magiques. Cette tourelle peut tirer **des munitions hextech** (des gemmes de magie instable de couleur bleu) et **des balles de shimmer** (des balles induits d'une substance chimique de couleur violette). Le but du joueur étant de descendre dans les profondeurs de Zaun, une ville sombre, en se frayant un chemin avec sa tourelle. Les munitions hextech détruisent les remparts hextech et les balles de shimmer détruisent les tuyaux de shimmer. En détruisant les obstacles, des **jauges se remplissent pour créer de nouvelles munitions**, afin d'aller de plus en plus vers les profondeurs. Si le joueur atteint le Puisard (la fin du niveau et le fin fond de la ville de Zaun), il gagne. S'il se retrouve à court de munitions, il perd.

Intention :

Le projet part d'un contexte **de création de jeu sur le framework Phaser 3.** Le langage de programmation est le **javascript** et le rendu se fait sur une page web. Le pitch du jeu a été élaborée après le visionnage de la **série Arcane**, qui contient un univers que j'ai fortement apprécié.

Objectifs

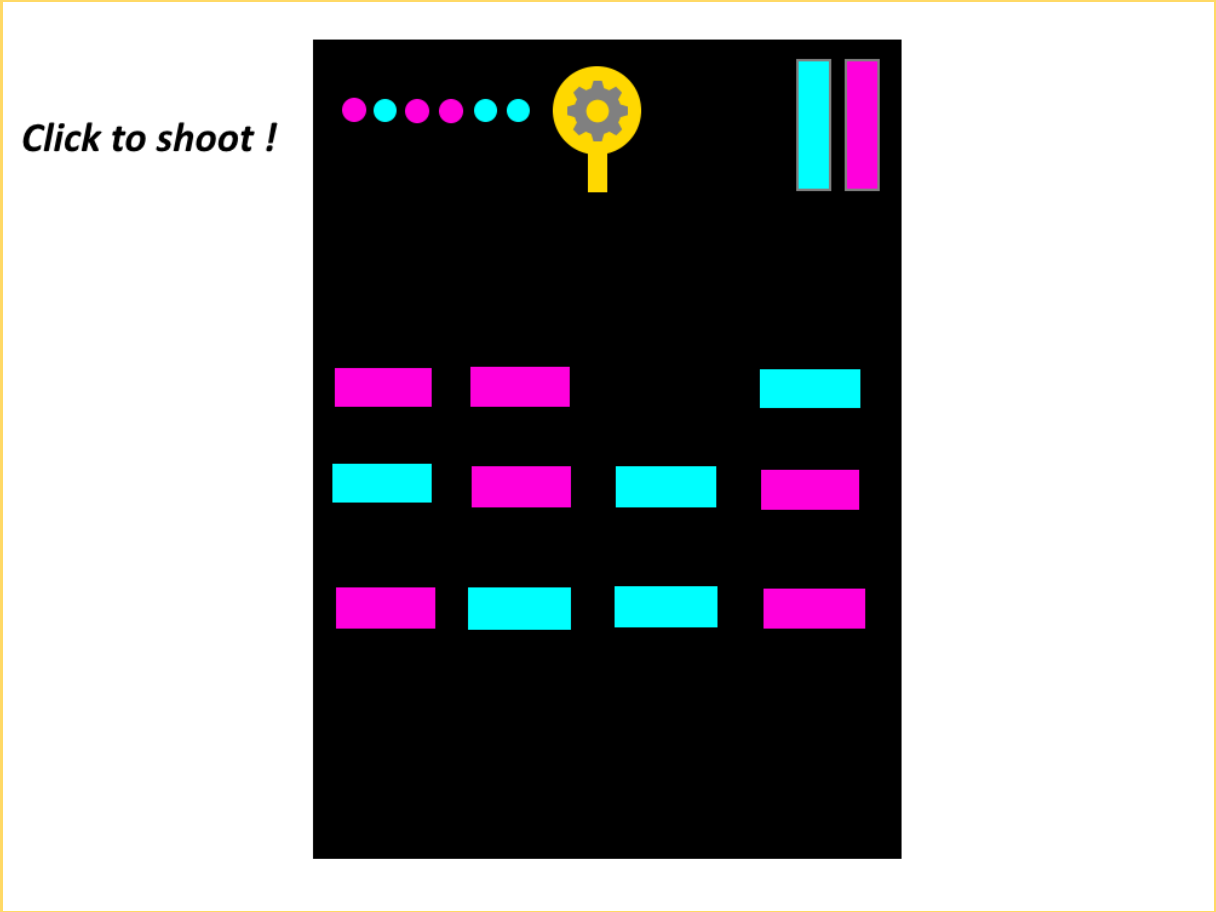
Le projet se poursuit sur **3 semaines**. Les livrables et l'exécutable seront disponibles sur un lien GitHub.

Les **missions** qui vont être effectuées lors de ce projet seront, de manière non exhaustive :

- Le joueur doit pouvoir diriger le canon de la tourelle avec la souris.
- Le joueur doit pouvoir tirer des munitions avec la tourelle.
- La tourelle doit être animée lors d'un tir de munition.
- Les munitions hextech doivent pouvoir détruire les obstacles hextech.
- Les munitions hextech doivent disparaître en touchant un obstacle shimmer.
- Les munitions shimmer doivent pouvoir détruire les obstacles shimmer.
- Les munitions shimmer doivent pouvoir détruire les obstacles hextech.
- La tourelle doit pouvoir descendre dans les profondeurs après l'élimination d'une rangée
- Un score est affiché, avec des feedbacks d'augmentation
- Les collisions entre une balle et un obstacle doit pouvoir générer des particules.
- Le jeu doit produire des sons de feedbacks et une musique d'ambiance.
- Une jauge hextech doit se remplir quand une munition hextech touche un obstacle hextech.
- Une jauge shimmer doit se remplir quand une munition shimmer touche un obstacle shimmer.
- Le niveau doit être décoré selon le thème de Zaun.
- Le joueur doit voir combien il lui reste de munitions.
- Le niveau doit comporter plusieurs obstacles.
- Les minutions à suivre s'affichent à gauche de la tourelle.
- Un écran de fin s'affiche quand il n'y a plus de munitions.

Des **missions secondaires** peuvent être effectuées, après les missions principales, s'il reste du temps :

- Certains obstacles sont constitués de shimmer et d'hextech, et peuvent être détruits avec n'importe quelle munition **(annulé pour cause de gameplay)**
- Des animations de Shader peuvent survenir pour des combos ou des feedbacks **(annulé pour cause de surcharge de feedbacks)**
- Certains obstacles comportent une rune, qui peuvent produire des effets d'augmentation du score, d'augmentation de jauges, d'augmentation des munitions ou de réduction du score
- Certains obstacles doivent subir plusieurs coups avant de se détruire **(annulé pour cause de rythme dans le jeu)**
- Des images html peuvent être affichées sur la page.
- Un menu de départ avec le bouton « jouer ».



3C

Personnage

Le joueur incarne une **tourelle**. Des animations lui sont attribuées :

- Quand le joueur change la position du canon (les engrenages tournent).
- Quand le joueur décide de tirer une munition (l'engrenage supérieure prend de la vitesse et le canon transmute de l'énergie chimique).

La tourelle se déplace vers le bas lorsque la rangée d'obstacles la plus proche est éliminée.

La tourelle peut tirer deux types de munitions :

- Les balles **hextech**, de couleur bleu, qui détruit les obstacles hextech. L'hextech, d'après l'univers d'*Arcane*, est une énergie magique contenue et catalysée dans un cristal.
- Les balles de **shimmer**, de couleur violette, qui détruit les obstacles de shimmer. Le shimmer, d'après l'univers d'*Arcane*, est une substance chimique capable d'altérer le corps humain et les sens une fois ingérée.

Caméra

La caméra est basée sur la **plateforme WEB**. Le niveau de jeu fait 950px de largeur sur 3000px de hauteur. Le joueur peut voir tout le niveau juste en scrollant la page web. Ce sera à lui de positionner verticalement la caméra pour avoir la vue qui lui conviendra le mieux. A savoir que la tourelle est toujours centrée horizontalement sur l'écran du joueur.

Contrôles

La canon change de rotation selon le curseur de la souris du joueur, en l'alignant d'après la position voulue. Le joueur clique sur le bouton gauche de la souris pour tirer un projectile.

Direction Artistique

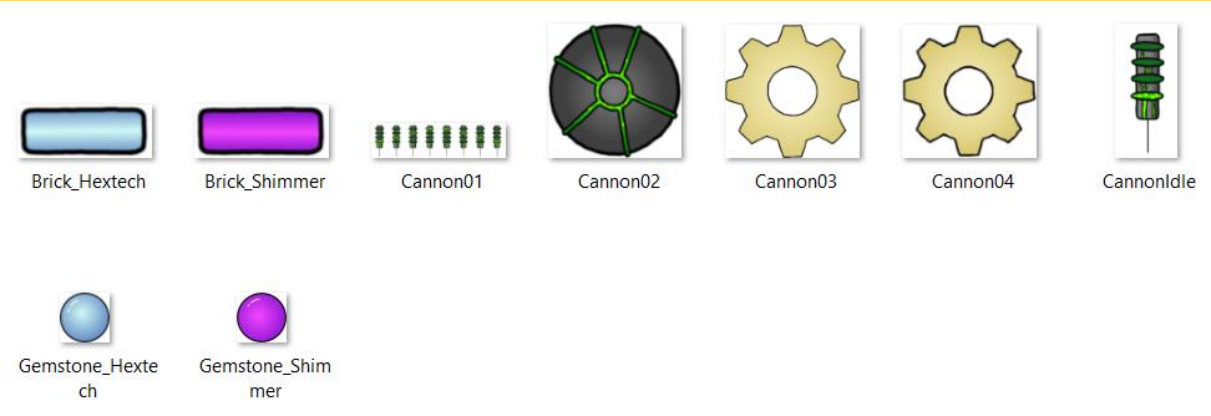
La direction artistique reprend celle de la **série Arcane**, produite par Riot Games.

Le côté sombre et industriel de Zaun ressort dans la pâte graphique, donnant une ambiance glauque au jeu.

Changelog

Lundi 22/11

- Brainstorm et élaboration d'un concept : **1H**
- Création de sprites : **3H**



- Utilisation de wamp et résolution de problèmes liés à l'affichages **: 1H**
- Apprentissage des structures, chargement d'images et placement des images sur la scène : **2H**
- Recherches et veilles techno : **1H**

Mardi 23/11

- Placement de quelques briques et du canon (apprentissage des couches et des transforms) : **1H**
- Animation de tir (apprentissage des timers) : **2H**

```
// ================= SHOOT ================================
this.input.on('pointerdown', function (pointer) {
  cannon.anims.play('shooting', 5, true);
  rotateGear = this.time.addEvent({ delay: 50, callback: TRotateGear, callbackScope: this, repeat: 70});
  this.cameras.main.shake(50, 0.005); // Camera Shake (duration, power)
  ShootHextech(pointer);
}, this);
// ========================================================
```

- Le canon tourne selon la position de la souris à l'écran (trigonométrie) : **2H**

```
// ================== ROTATE CANNON ==========================
this.input.on('pointermove', function (pointer) {

    var dist = Phaser.Math.Distance.BetweenPoints(cannon, pointer);
    var adj = cannon.x - pointer.x;
    var cos = adj / dist;
    var degree = Phaser.Math.RadToDeg(cos) + 180;
    cannon.setAngle(degree);

    var dist2 = Phaser.Math.Distance.BetweenPoints(cannon, pointer);
    var adj2 = pointer.x - cannon.x;
    var cos2 = adj2 / dist2;
    var degree2 = Phaser.Math.RadToDeg(cos2) + 180;
    gearBottom.setAngle(degree2);
}, this);
// ============================================================
```

- Le canon tir un projectile selon la position de la souris (trigonométrie et vélocité) : **2H**

```
function ShootHextech(pointer)
{
    var hextechBall = hextechBalls.create(posX, posY, 'ball_hextech');
    hextechBall.setBounce(1);
    hextechBall.setCollideWorldBounds(true);
    var vectorVelocity = new Phaser.Math.Vector2(pointer.x - 450, pointer.y);
    vectorVelocity.normalize();
    hextechBall.setVelocity(vectorVelocity.x * power, vectorVelocity.y * power);
}
```

Mercredi 24/11

- Mise en place et écriture du document de GD : **3H**
- Les projectiles sont instanciés au bout du canon, et non sur la tourelle (théorème de Pythagore et trigonométrie) : **1H30**

```
// Calculate start position of balls when shoot
function CalculateBallPos(pointer)
{
    let distCannon = 147;
    let dist = Phaser.Math.Distance.BetweenPoints(cannon, pointer);
    let adj = cannon.x - pointer.x;
    let cos = adj / dist;
    let posX = 450 - cos * distCannon;
    let posY = 100 + Math.sqrt((distCannon * distCannon) - ((cos * distCannon) * (cos * distCannon)));
    let pos = new Phaser.Math.Vector2(posX, posY);

    return pos;
}
```

- Les particules hextech sont instanciées lors d'un tir de balles hextech : **3H**

```
// Spawn hextech particles near the end of the cannon
function SpawnHextechParticlesShoot(posX, posY, ptc)
{
  emitter = ptc.createEmitter();

  emitter.setPosition(posX, posY);
  emitter.setSpeed(200);
  emitter.setBlendMode(Phaser.BlendModes.ADD);
  emitter.setQuantity(10);
}
```

```
// TIMER FUNCTION
// Stop particles after delay when cannon shots a bullet
function TStopParticlesShoot()
{
  emitter.stop();
}

// TIMER FUNCTION
// Reduce the alpha of particle sprite
function TReduceParticleAlpha()
{
  emitter.setAlpha(emitter.alpha.propertyValue - 0.02);
}
```

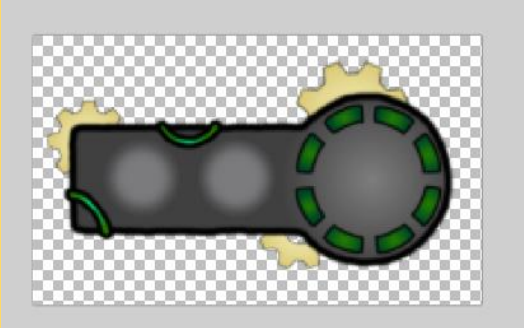
Lundi 29/11

- Les balles Hextech explosent les blocks hextech à leur contact avec une animation de particules. Les balles Hextech sont détruites au contact d'un block shimmer, et augmentent la taille de ce dernier. **3H**

```
// Hextech ball hit hextech brick
function HitHextech(hextechBall, hextechBlock)
{
  var particlesHextechDestroy = this.add.particles('ptcHextechDestroy');
  SpawnHextechParticlesDestroy(hextechBlock.x, hextechBlock.y, particlesHextechDestroy)

  animDestroyHextechBlock = this.time.addEvent({ delay: 1, callback: TAnimDestroyHextechBlock, args: [hextechBlock], callbackScope: this, repeat: 200});
  dispawnHextechBlock = this.time.addEvent({ delay: 201, callback: TDispawnHextechBlock, args: [hextechBlock], callbackScope: this});

  reduceParticleAlphaDestroyHextech = this.time.addEvent({ delay: 1, callback: TReduceParticleAlphaDestroyHextech, callbackScope: this, repeat : 50});
  stopParticlesDestroyHextech = this.time.addEvent({ delay: 51, callback: TStopParticlesDestroyHextech, callbackScope: this});
}


// Hextech ball hit shimmer brick
function FailHextech(hextechBall, shimmerBlock)
{
  shimmerBlock.setScale(shimmerBlock.scaleX + 0.1, shimmerBlock.scaleY + 0.1);
  hextechBall.disableBody(true, true);
}
```

- Conception et graphisme d'un panneau de chargeur pour indiquer au joueur quelle munition est chargée. **2H30**

```
// ================= ANIM : LOADER PANEL ========================
panelLoader = this.physics.add.sprite(150, 50, 'panelLoaderBullets');

this.anims.create({
  key: 'loader',
  frames: this.anims.generateFrameNumbers('panelLoaderBulletsAnim', { start: 0, end: 7 }),
  frameRate: 32,
  repeat: -1
});

panelLoader.anims.play('loader', 5, true);
// ============================================================
```



- Implémentation du chargeur en jeu (données et visuels) : **1H**
- La tourelle peut tirer des balles de shimmer (fonctionnelles avec les obstacles) : **1H**

Mardi 30/11

- Affichage des munitions restantes avec une police différente: **2H**

```
// ================= SETUP LOADER ============================
loader01 = 0;
loader02 = 1;
loader03 = 0;

loaderBallHextech03 = this.add.image(98, 52, 'ballHextech').setScale(0.5);
loaderBallHextech02 = this.add.image(140, 52, 'ballHextech').setScale(0.5).setVisible(false);
loaderBallHextech01 = this.add.image(200, 52, 'ballHextech');
loaderBallShimmer03 = this.add.image(98, 52, 'ballShimmer').setScale(0.5).setVisible(false);
loaderBallShimmer02 = this.add.image(140, 52, 'ballShimmer').setScale(0.5);
loaderBallShimmer01 = this.add.image(200, 52, 'ballShimmer').setVisible(false);

remainingMun = 30;
txtMun = this.add.text(100, 120, 'X ' + remainingMun, { fill: '#ffffff' });
txtMun.align = 'center';
txtMun.setFontSize(40);
txtMun.setFontFamily('font1');
// ============================================================
```

- Correction de bugs sur particules : **1H**
- Mise en place de sons et de la musique principale : **1H**

- Les jauges se remplissent lors d'une explosion d'obstacles + animation des jauges : **2H**

```
// ================= BARS ================================
barHextech01 = this.physics.add.sprite(300, 150, 'barHextech01').setOrigin(0.5, 1).setScale(1, 0.5);

this.anims.create({
  key: 'barHextechAnim',
  frames: this.anims.generateFrameNumbers('barHextechSheet', { start: 0, end: 7 }),
  frameRate: 32,
  repeat: -1
});

barHextech01.anims.play('barHextechAnim', 5, true);

barHextech02 = this.add.image(300, 50, 'bar02');

barShimmer01 = this.physics.add.sprite(600, 150, 'barHextech01').setOrigin(0.5, 1).setScale(1, 0.5);

this.anims.create({
  key: 'barShimmerAnim',
  frames: this.anims.generateFrameNumbers('barShimmerSheet', { start: 0, end: 7 }),
  frameRate: 32,
  repeat: -1
});

barShimmer01.anims.play('barShimmerAnim', 5, true);

barShimmer02 = this.add.image(600, 50, 'bar02');

valueHextechBar = 50;
valueShimmerBar = 50;

addOne = this.add.image(210, 140, 'addOne');
addOne.setAlpha(0);
// ========================================================
```

- Ajout d'une animation lorsque le joueur récupère une munition supplémentaire en remplissant une des jauges : **1H**
- Modification des sons de feedback : **1H**

Mercredi 01/12

- Mise en place d'un score et ses feedbacks : **1H**

```
// ================== SCORE ===================================
actualScore = 0;
txtScore = this.add.text(740, 70, actualScore, { fill: '#ffffff' });
txtScore.align = 'center';
txtScore.setFontSize(40);
txtScore.setFontFamily('font1');
isTextScoreOnAnim = false;
// ============================================================
```

- Réalisation des données des rangées d'obstacles pour les faire avancer : **2H**

```
// TIMER FUNCTION
// Move blocks with times
function TMoveBlocks()
{
 for (var i = 0; i < hextechBlocks.children.entries.length; i++)
 {
   hextechBlocks.children.entries[i].y -= 2;
   hextechBlocks.children.entries[i].refreshBody();
 }

 for (var j = 0; j < shimmerBlocks.children.entries.length; j++)
 {
   shimmerBlocks.children.entries[j].y -= 2;
   shimmerBlocks.children.entries[j].refreshBody();
 }

 bg.y -= 2;
 bgCollider.y -= 2;
 bgCollider.refreshBody();
}


function CheckRow()
{
 var tempCanApproach = 1;
 var tempArray = rowRows[numberRow];

 for (var i = 0; i < tempArray.length; i++)
 {
   if(tempArray[i].name != "isDead")
   {
     tempCanApproach = 0;
   }
 }

 return tempCanApproach;
}
```

- Level Design complété : tous les obstacles sont posés : **2H**
- Mise en place du décor : **2H**
- Ajout de rangées d'obstacles : **1H**

Lundi 06/12

- Panneau de fin de partie avec le nombre de score indiqué : **2H**
- Panneau de début de partie avec « Press Space to Play » : **2H**

```
// ================= END PANEL ============================
bulletsInGame = 0;
isEnd = false;

bgBlack = this.add.image(450, 1500, 'bgBlack');

endPanel = this.add.image(450, 400, 'endPanel').setVisible(false).setScale(0.6);

txtEndScore = this.add.text(500, 452, actualScore, { fill: '#ffffff' });
txtEndScore.align = 'center';
txtEndScore.setFontSize(50);
txtEndScore.setFontFamily('font1');
txtEndScore.setVisible(false);

startPanel = this.add.image(450, 400, 'startPanel').setScale(1.2);

txtPlay = this.add.text(235, 405, "Press Space to Play", { fill: '#ffffff' });
txtPlay.setFontSize(50);
txtPlay.setFontFamily('font1');
decreaseAlphaText = this.time.addEvent({ delay: 5, callback: TDecreaseAlphaText, args: [this], callbackScope: this, repeat: 20});
stateAlphaText = 1;
// ========================================================
```

- Design et intégration d'images dans la page html pour faire comprendre au joueur des infos : **2H**
- Design et intégration des runes : **2H**
- Animation des runes **: 1H**

FIN DU PROJET

Le mardi et le mercredi suivants seront consacrés au projet master.