Bachelor-1 GDA

In No Time !

Document Technique - Projet de production Bachelor 1 Game Design,
ETPA Rennes 2019-2020

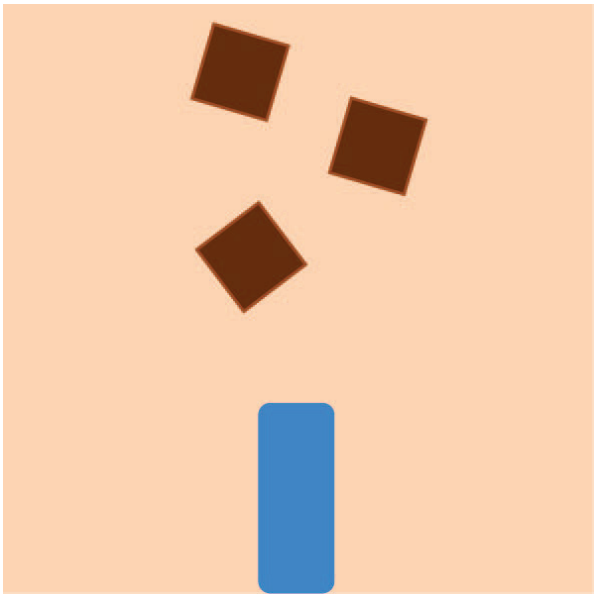
MARIETTE Guillaume

10/06/2020

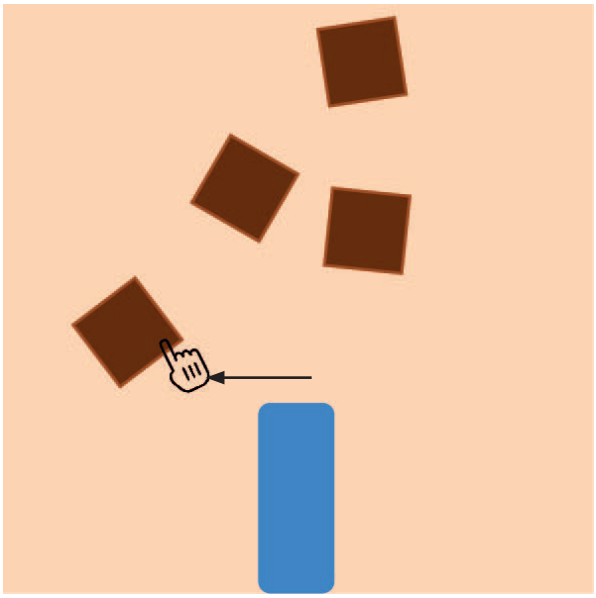
Push Out

Il s'agit d'un jeu d'action - réflexe. Des objets tombent sur un personnage et le joueur doit donc pousser ces derniers hors de l'écran avant qu'ils ne touchent le personnage. La difficulté étant que la vitesse des objets qui tombent augmente plus ont réussi de micro games.

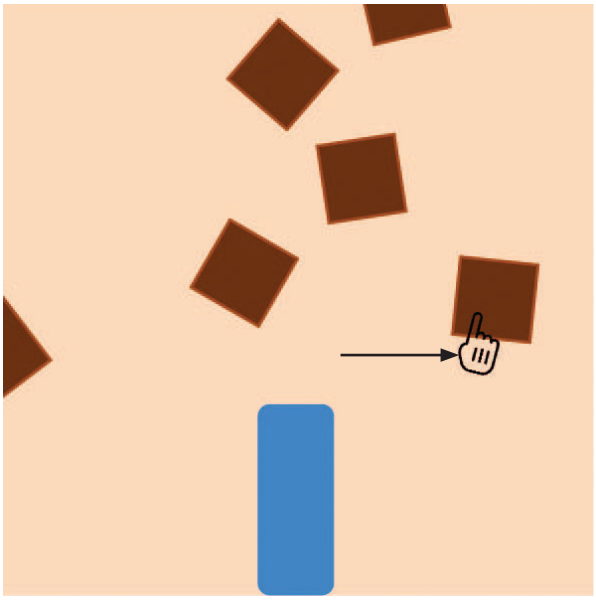
Gameplay

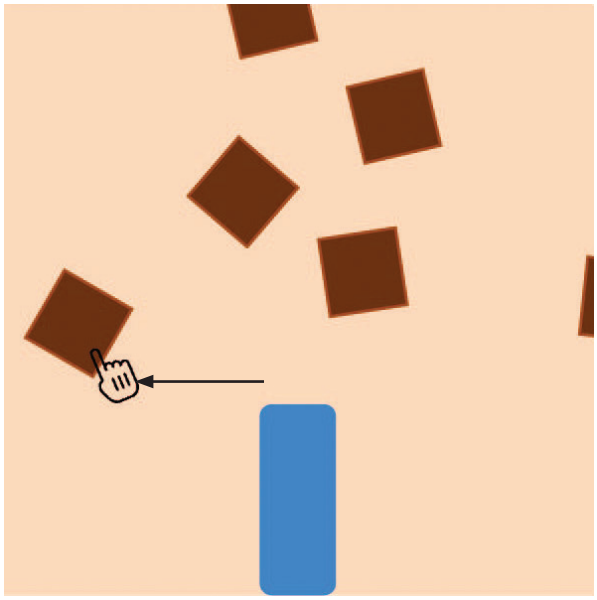


Des objets tombe sur un personnage



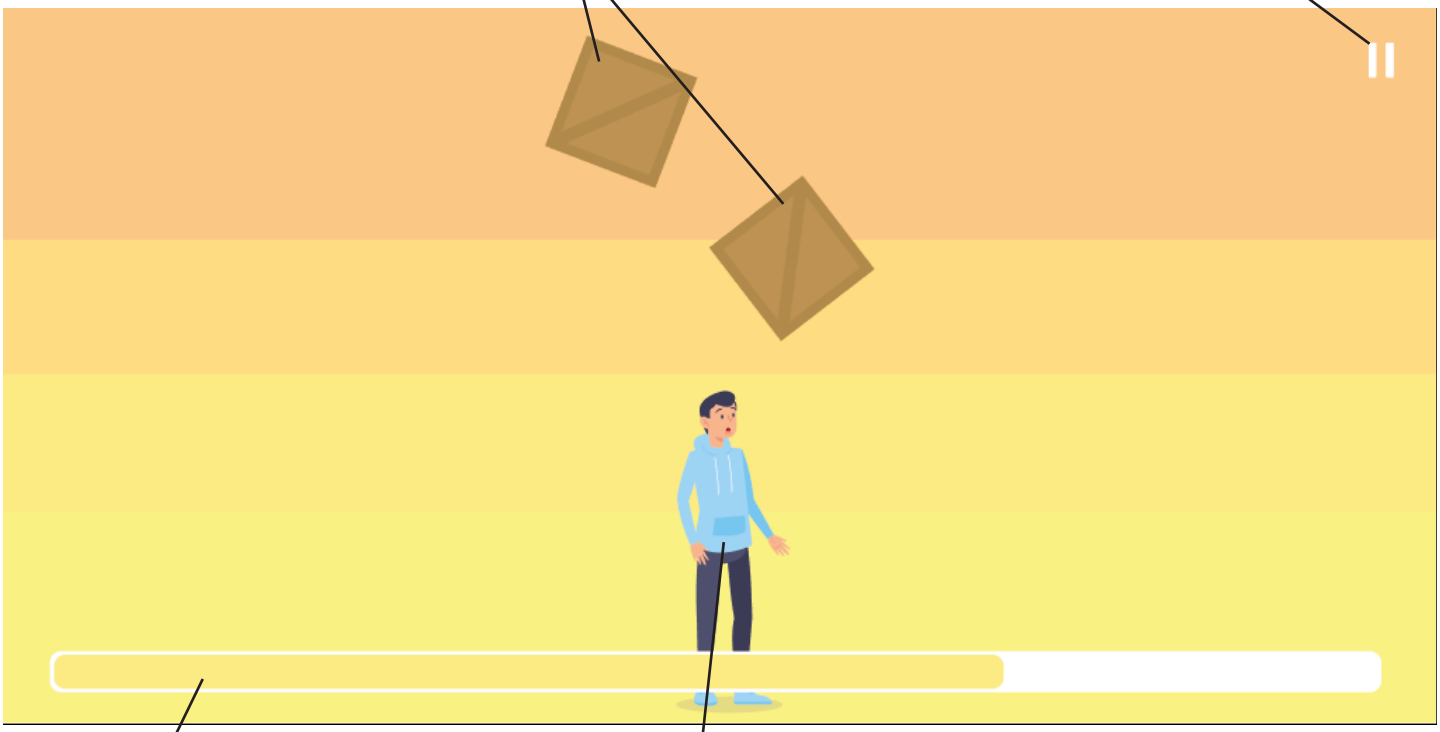
Le joueur pousse l'un des objet hors de l'écran





Et ainsi de suite jusqu'a la fin du micro games

Maquette



Art

Fond
Flat Design
Différentes teintes d'Orange
Style dégradé

Personnage Objectif
Flat Design
Contraste avec le fond

Chronomètre
Flat Design

Contour blanc et teinte clair pour la visibilité

Données

Nous allons voir les variables et fonctions principales.

Les objets qui tombent : nous avons intégré une gravité propre à la scène, puis insérer des objets hors de l'écran, au-dessus du personnage. À l'ajout d'une physique aux objets, ces derniers tombent touche le personnage.

```
class Scene1 extends Phaser.Scene{
    constructor(){
        super({
            key:'Scene1',
            physics: {
                default: 'arcade',
                arcade: {
                    gravity: { y: 300 }
                }
            }
        });
}
```

```
//caisse 1

this.caisse = this.physics.add.sprite(960, -500, 'caisse').setScale(0.25);
this.physics.add.collider(this.caisse,this.homme2,hitHomme, null, this);
```

Attraper les objets : pour cela, chaque objet devait être indépendant des autres. Nous avons donc appliqué cette fonction à chacun d'entre eux.

```
this.caisse.on('drag', function (pointer, dragX, dragY) {

    this.x = dragX;
    this.y = dragY;
});
```

Victoire ou **Défaite** : un objet invisible tombe à la même vitesse que les objets à éviter, mais de plus haut et quand celui-ci touche le personnage après que tous les objets ont été éviter, le jeu affiche victoire. Pour la défaite, c'est un peu pareil, sauf que ce sont les objets à éviter qui l'enclenchent.

Victoire :

```
this.fallwin = this.physics.add.sprite(2721, -1600, 'sol');
this.physics.add.overlap(this.homme2, this.fallwin, WinPass, null, this);


function WinPass(homme2, fallwin){
    this.winText = this.add.text(960,540, 'Win !', {fontSize: '100px', fill:'#000'}).setOrigin(0.5);
    this.score += 100;
```

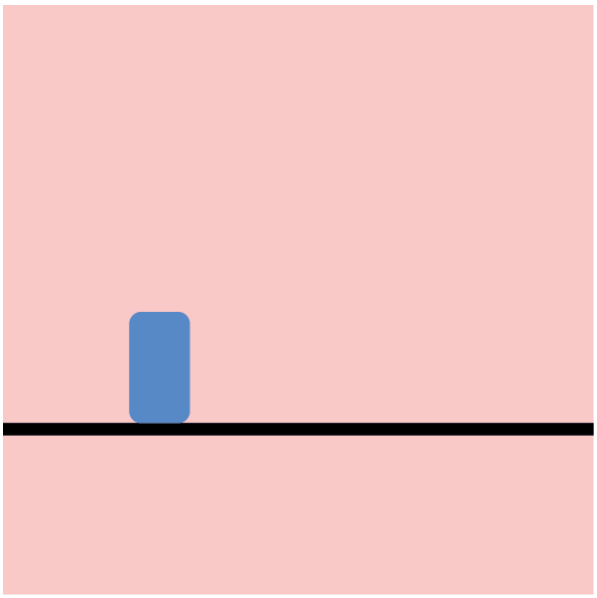
Défaite :

```
function hitHomme(homme2, caisse){
    this.vie --;
    this.niv = 2;
    this.deathText = this.add.text(960,540, 'Perdu !', {fontSize: '100px', fill:'#000'}).setOrigin(0.5);
    this.physics.pause();
    this.gameOver=true;
```

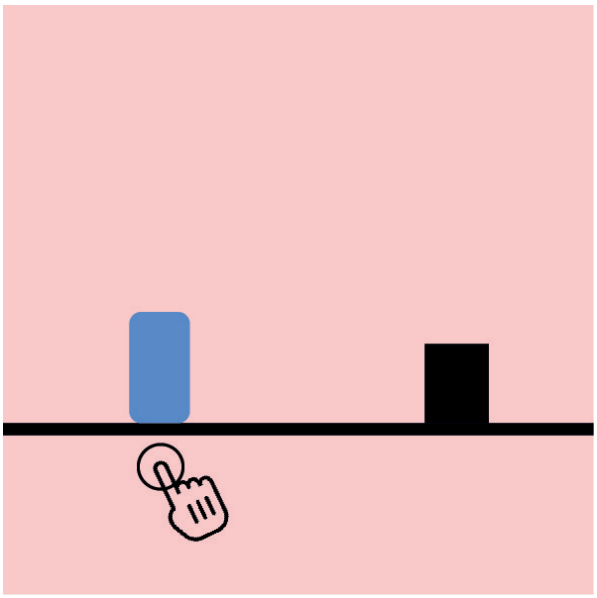
Jump Guy

Il s'agit d'un jeu d'action - réflexe. Le joueur contrôle un personnage sur un skate. Le personnage avance tout seul et le joueur doit appuyer au bon moment pour sauter par-dessus les obstacles qui sont devant lui. Plus le temps passe plus la difficulté monte en augmentant la vitesse du skate et le nombre d'obstacles.

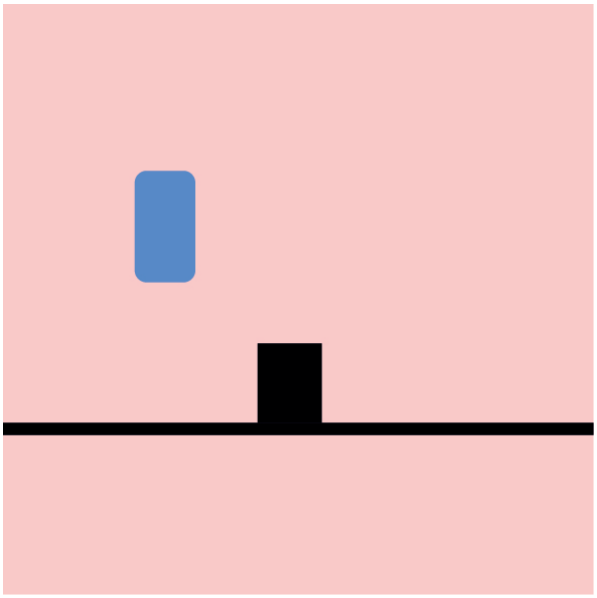
Gameplay

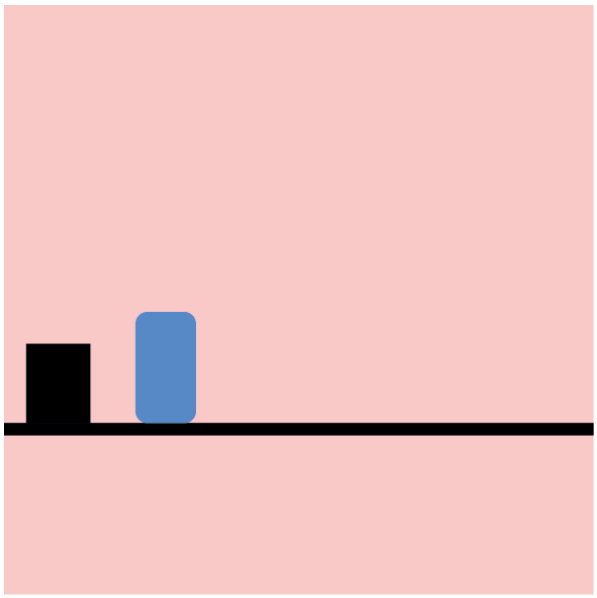


Le personnage avance tout seul vers la droite



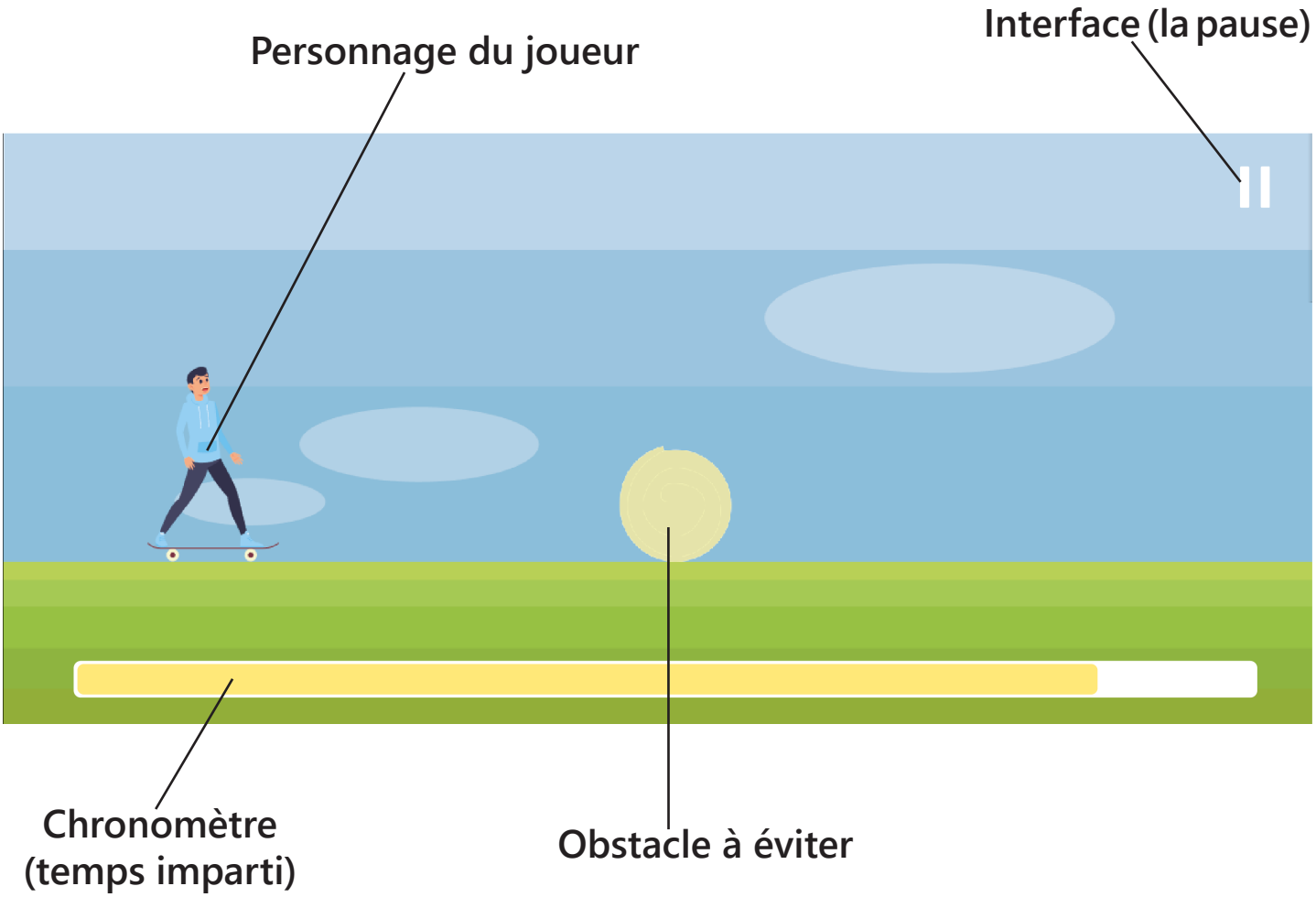
Le joueur appuie sur le personnage



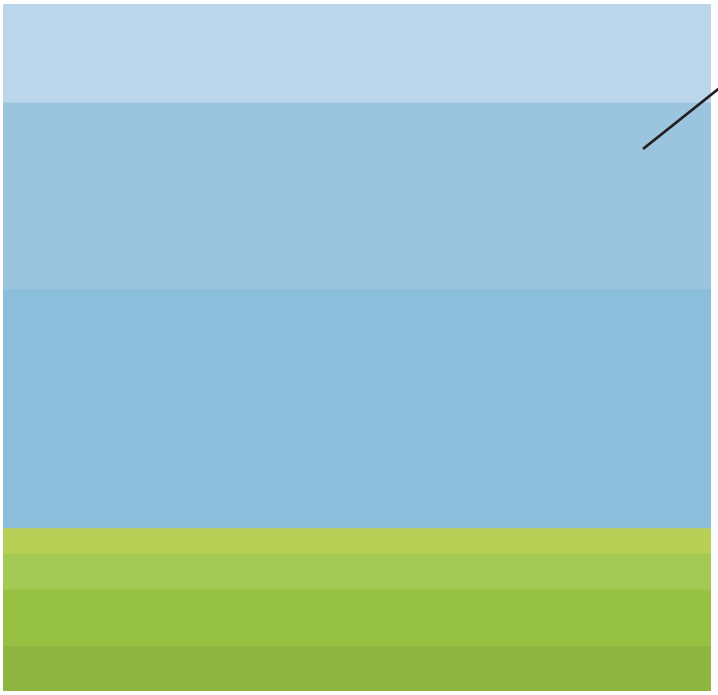


Le personnage saute et passe au-dessus de l'obstacle

Maquette



Art



Fond
Flat Design
Différentes teintes de Bleu et de Vert
Style dégradé

Données

Nous allons voir les variables et fonctions principales.

Le saut : cette scène utilise la gravité principale du jeu. Grâce à la condition, «if», le jeu vérifie si le joueur à appuyer sur le personnage et si ce dernier touche le sol. Si oui le personnage se voit attribuer une vélocité Y négative, le faisant sauté.

```
    default: 'arcade',
    arcade: {
        gravity: { y: 1300 },
        debug: true
    }
}
```

```
this.player.on("pointerup",()=>{
    if (this.player.body.touching.down) {
        this.player.setVelocityY(-1000);
        this.player.anims.play('jump', true);
```

Victoire ou **Défaite** : un objet invisible à la vélocité X négative se dirige vers le joueur et quand celui-ci touche le personnage après que tous les objets aient été éviter, le jeu affiche victoire. Pour la défaite, c'est l'objet à éviter qui l'enclenchent.

Victoire :

```
this.win = this.physics.add.sprite(3800, 540, 'win').setImmovable(true);
this.win.body.setAllowGravity(false);
this.win.setVelocityX(-900);
this.physics.add.overlap(this.player, this.win, WinPass, null, this);
```

```
function WinPass(Player, Win){
    this.winText = this.add.text(960,540, 'Win !', {fontSize: '100px', fill:'#000'}).setOrigin(0.5);
```

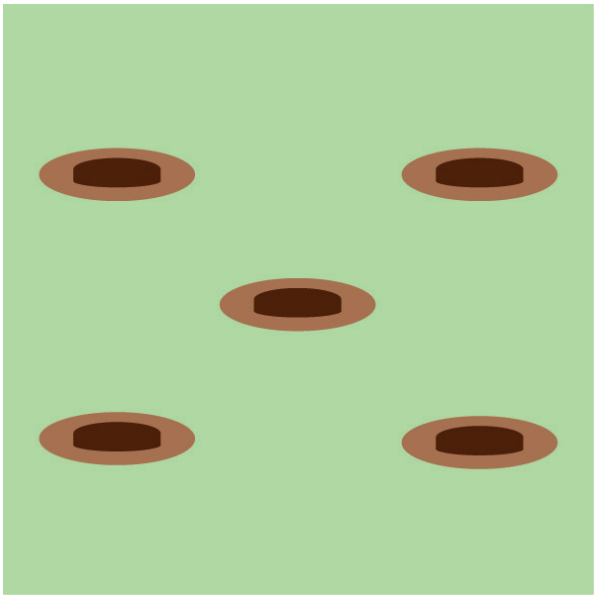
Défaite :

```
function hitPlayer(Player, Paille){
    this.vie = 2;
    this.niv = 1;
    this.deathText = this.add.text(960,540, 'Lose !', {fontSize: '100px', fill:'#000'}).setOrigin(0.5);
```

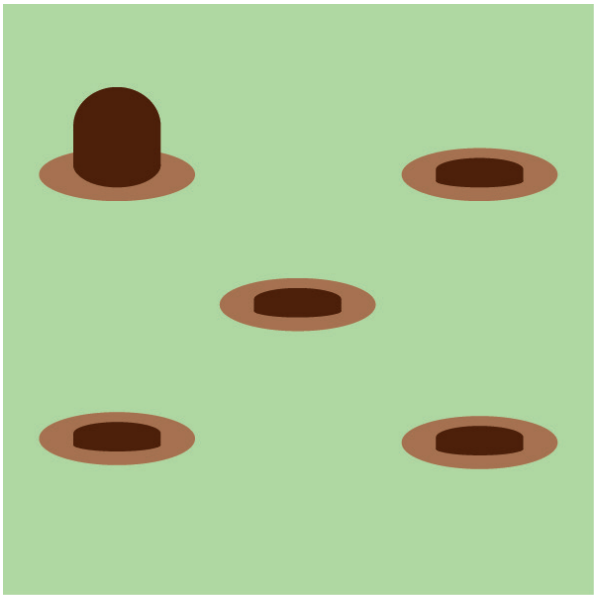
Hit Them

Il s'agit d'un jeu d'action - réflexe. Des taupes sortent de terre aléatoirement et le joueur doit toutes leur taper dessus avant la fin du temps impartis. Plus le temps passe plus le nombre de taupes augmente et leur temps d'apparition diminue.

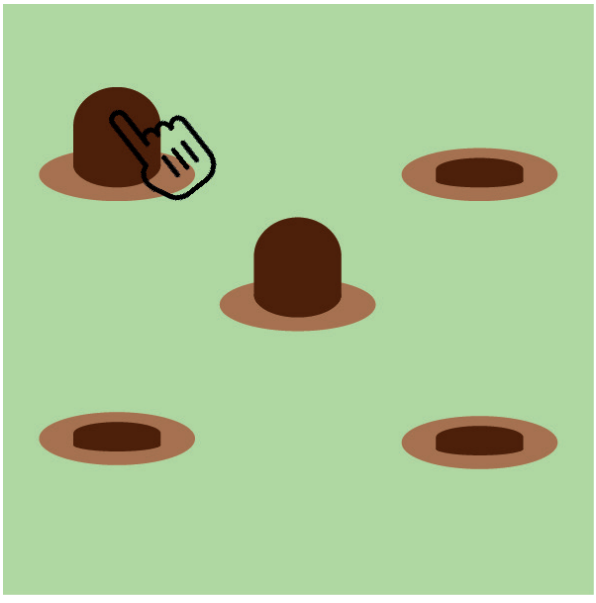
Gameplay

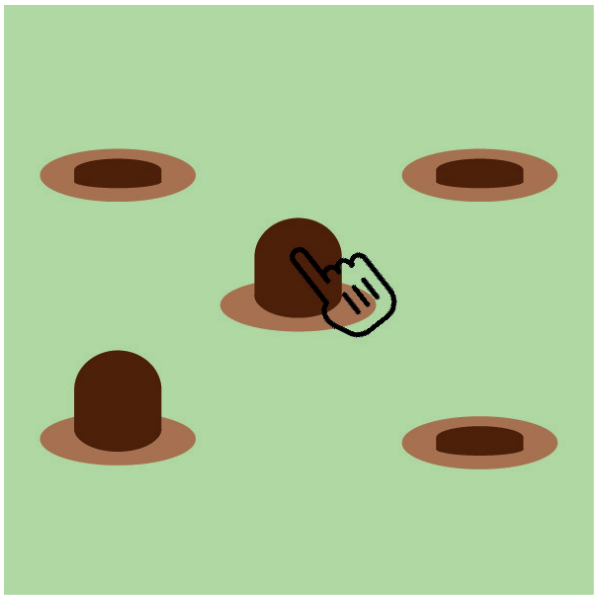


Les trous de taupe sont mis en évidence



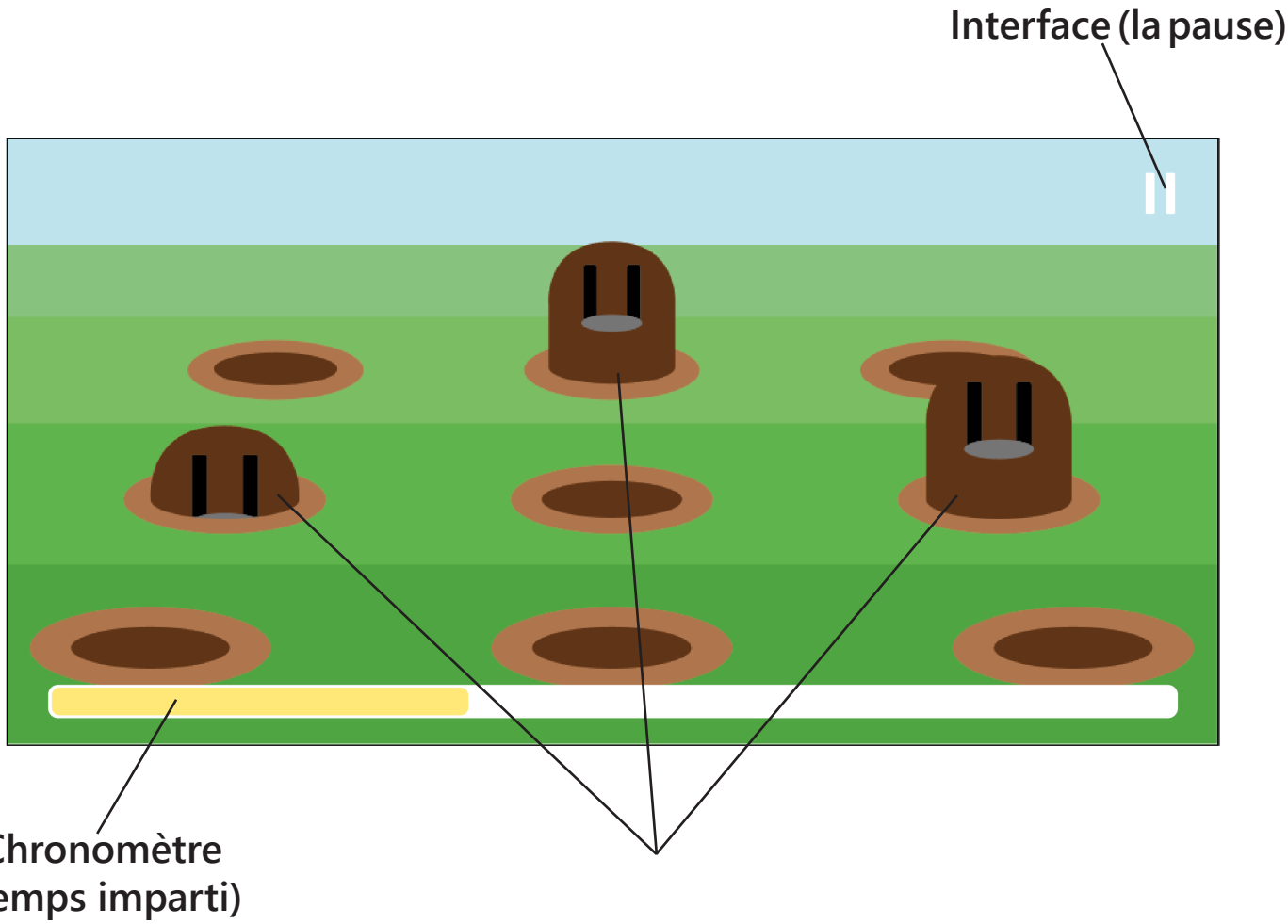
Une taupe apparaît



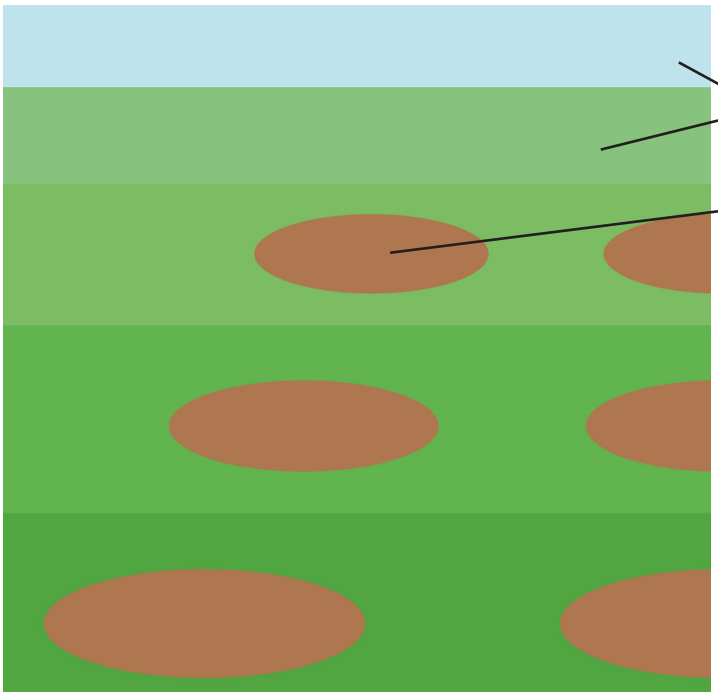


Le joueur tape dessus et ainsi de suite

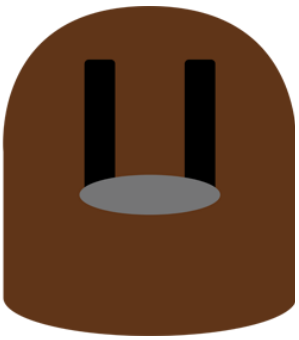
Maquette



Art



Fond
Flat Design
Différentes teintes de vert +
une teinte de bleu ciel
Style dégradé
Trou de taupe couleur terre



Taupe
Flat Design
Mignon

Données

Nous allons voir les variables et fonctions principales.

L'apparition des taupes : les taupes sont présentes au lancement du jeu, mais grâce à un «time event», après un certain délai, une animation de sortie de terre les font apparaître à l'écran.

```
//ligne taupe millieu
this.taupe4 = this.physics.add.sprite(1392,394, 'taupeS').setOrigin(0,0);
this.taupe4.setInteractive();
this.taupe5 = this.physics.add.sprite(836,394, 'taupeS').setOrigin(0,0);
this.taupe5.setInteractive();
this.taupe6 = this.physics.add.sprite(280,394, 'taupeS').setOrigin(0,0);
this.taupe6.setInteractive();
```

```
    this.time.addEvent({
    delay: 1200,
    callback: ()=>{
        this.taupe5.anims.play('taupanim', true);

    },
    loop: false
})
```

Clique sur les taupes : tout d'abord, nous avons rendu les sprites de taupe interactif, grâce à la fonction «setInteractive()». Puis nous avons ajouté une condition à chaque taupe qui sort de terre. Lorsqu'elles sont cliquées, une animation de rentrer en terre se lance, puis le compteur de taupe s'ajoute 1.

```
this.taupe4 = this.physics.add.sprite(1392,394, 'taupeS').setOrigin(0,0);
this.taupe4.setInteractive();


this.taupe1.on("pointerup",()=>{
this.taupe1.anims.play('tauprevers', true);
this.tp ++;
console.log("Tp");
})
```

Victoire ou **Défaite** : grâce à un compteur, au bout d'un certain nombre de taupes toucher, la victoire s'enclenche. Pour la défaite, un compte à rebours est intégré grâce à un «time event».

Victoire :

```
if (this.tp == 1) {
    this.winText = this.add.text(960,540, 'Win !', {fontSize: '100px', fill:'#000'}).setOrigin(0.5);
    this.score += 100;
```

Défaite :

```
this.time.addEvent({
delay: 5500,
callback: ()=>{
   this.vie --;
    this.niv = 3;
    this.deathText = this.add.text(960,540, 'Lose !', {fontSize: '100px', fill:'#000'}).setOrigin(0.5);
    this.physics.pause();
    this.gameOver=true;
```

Plane Bounce

Il s'agit d'un jeu d'action - réflexe. Le joueur doit faire rebondir un avion lorsqu'il commence à piquer du nez. Plus le temps passe plus la difficulté monte en augmentant la vitesse de l'avion et en faisant apparaître des obstacles.

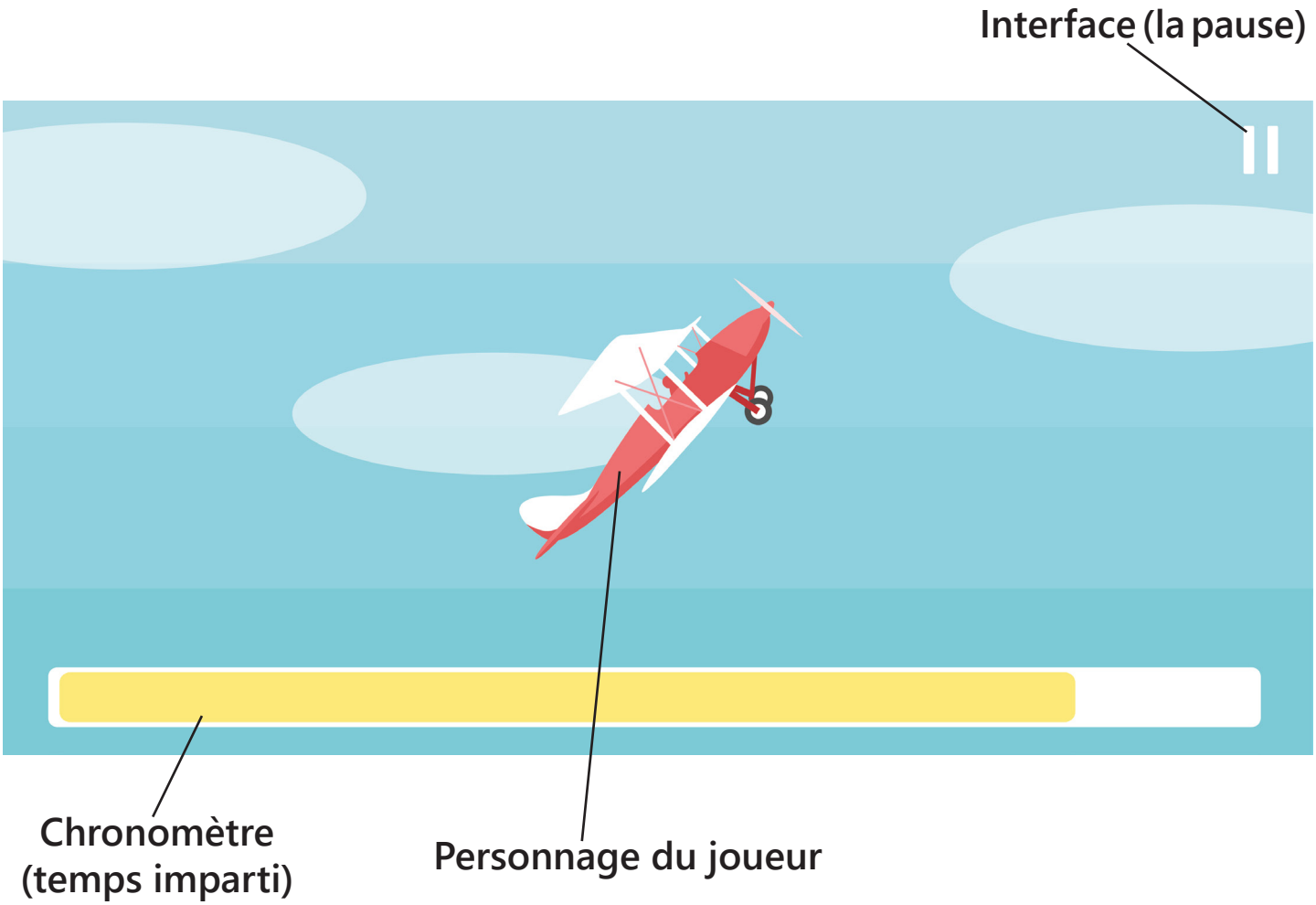
Gameplay

On dirige un avion en vol

L'avion pique du nez

Le joueur appuie sur l'avion, puis celui-ci rebondi et ainsi de suite

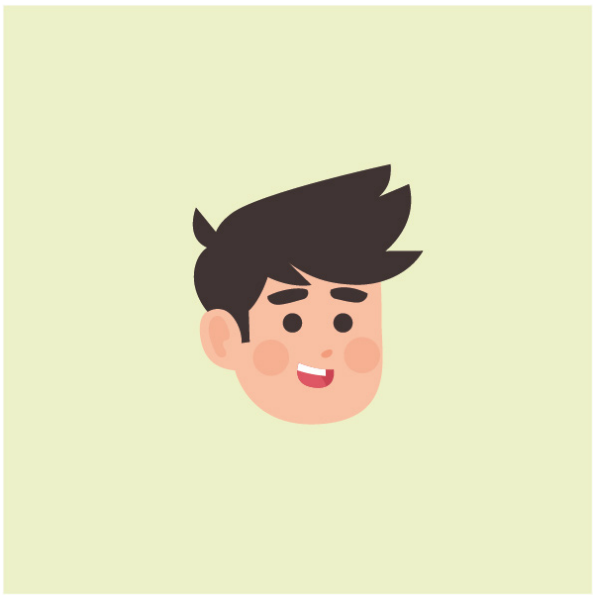
Maquette


Interface (la pause)
Chronomètre
(temps imparti)
Personnage du joueur

Simon Face

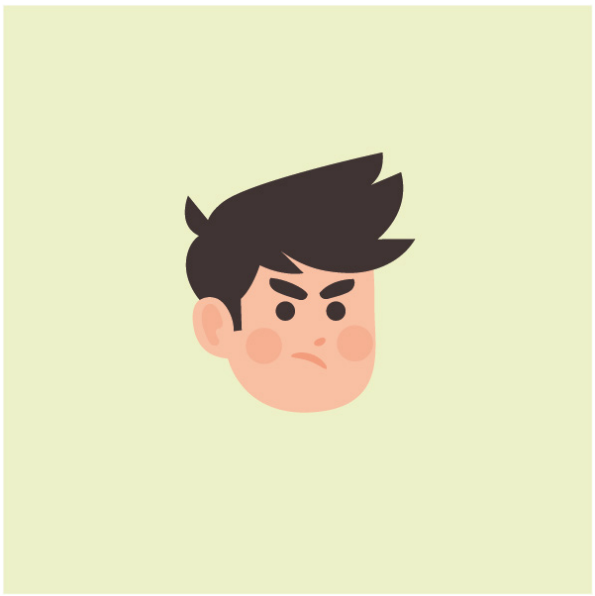
Il s'agit d'un jeu d'action - réflexion. Un visage apparaît à l'écran et change d'expression plusieurs fois. Puis une sélection d'expression apparaît et le joueur doit choisir celles qui sont apparues précédemment dans le bon ordre. Plus le temps passe plus la difficulté monte en augmentant le nombre d'expressions.

Gameplay

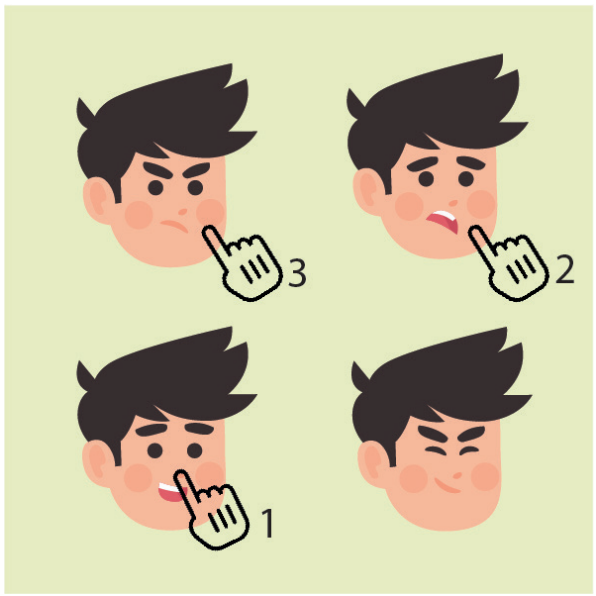


Une visage apparaît

Le visage change d'expression

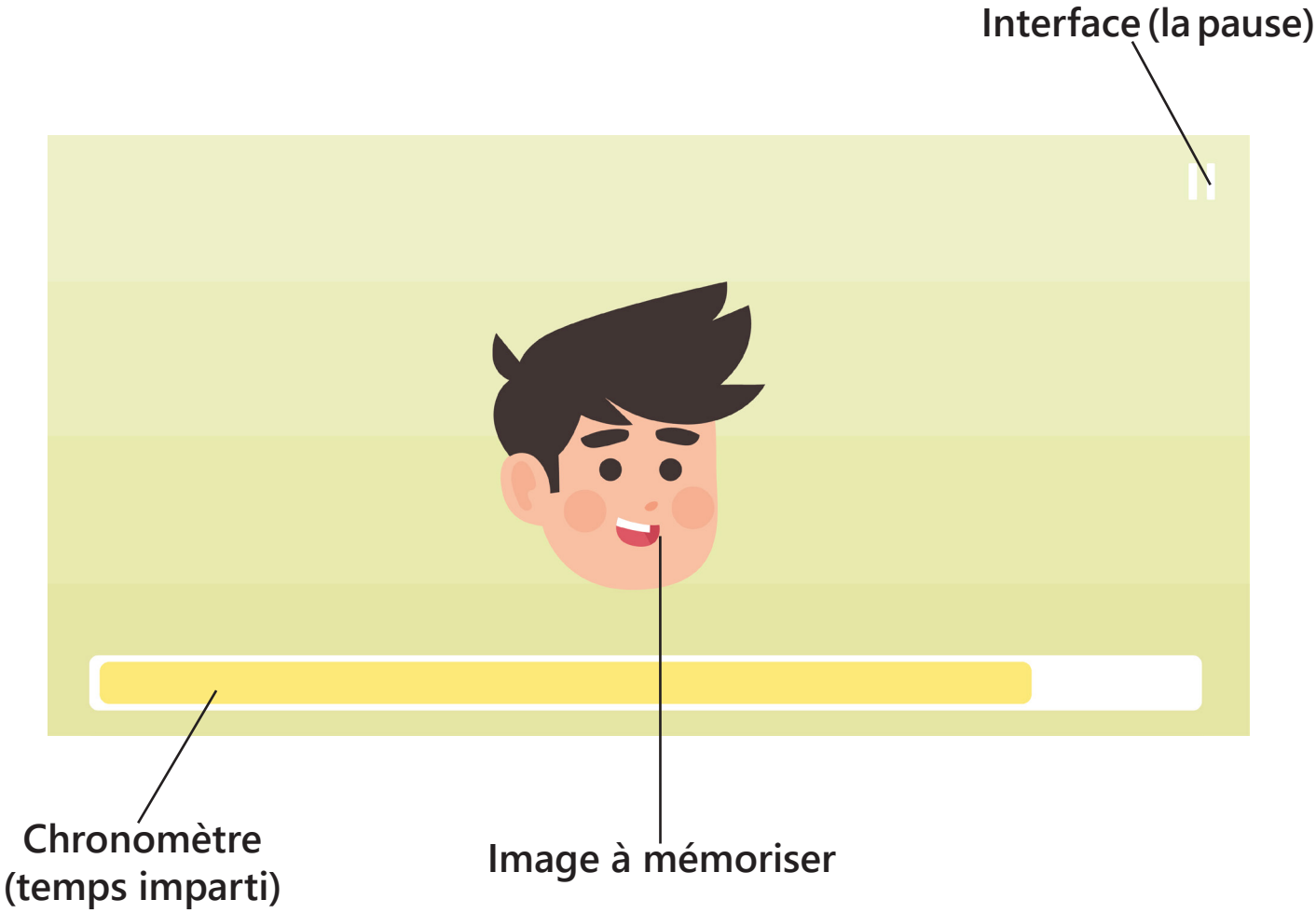


Le visage change d'expression une nouvelle fois



Joueur doit se souvenir des expressions apparues

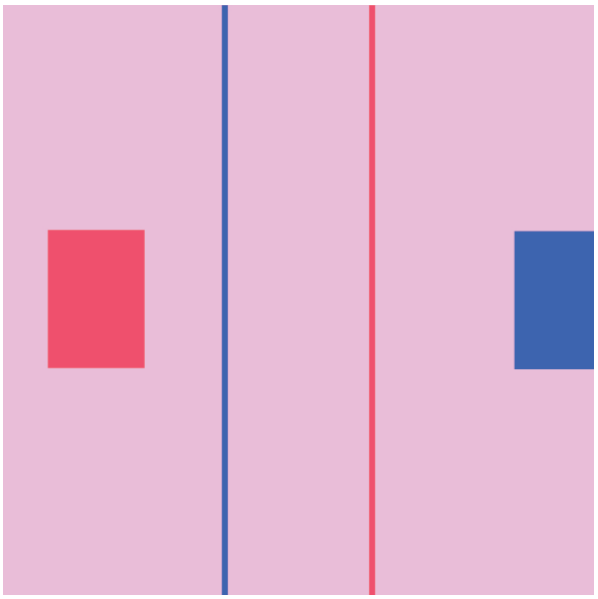
Maquette


Interface (la pause)
Chronomètre
(temps imparti)
Image à mémoriser

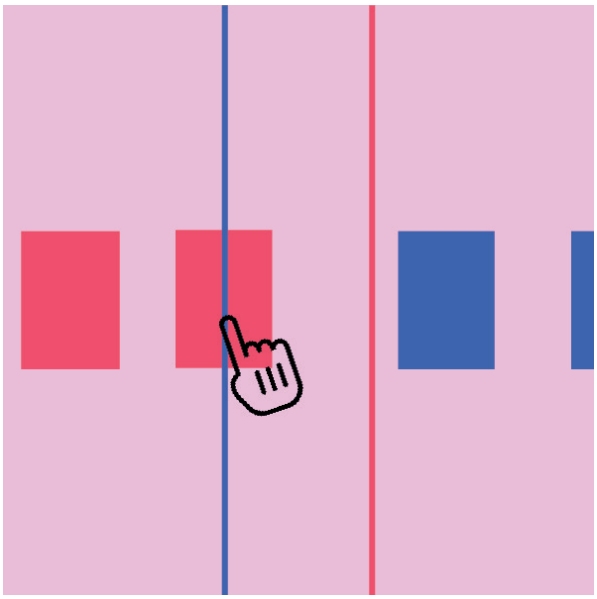
Melody Maniac

Il s'agit d'un jeu d'action - réflexe. Des notes arrivent des deux cotés de l'écran et le joueur doit appuyer dessus au bon moment et en rythme avec une petite mélodie. Plus le temps passe plus la difficulté monte en augmentant la vitesse des notes.

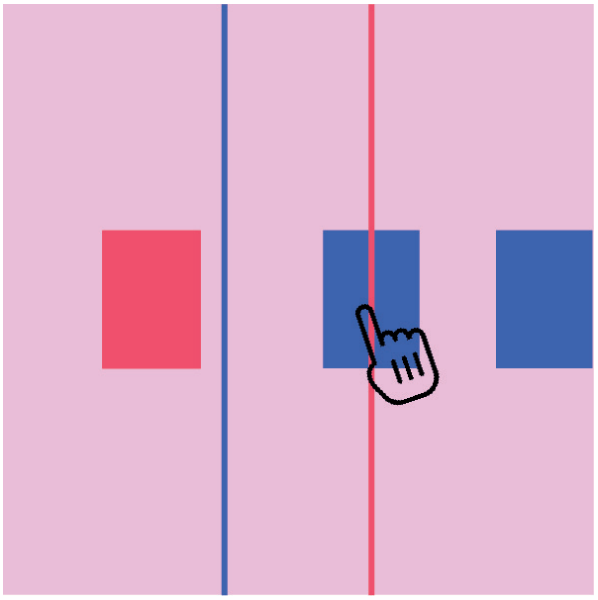
Gameplay



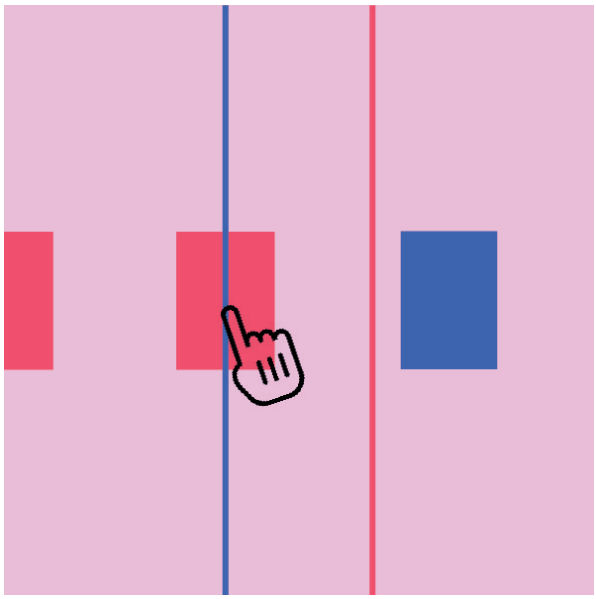
Des nôtes arrivent des deux coté de l'écran



Le joueur appuie sur la nôte de gauche, au bon moment

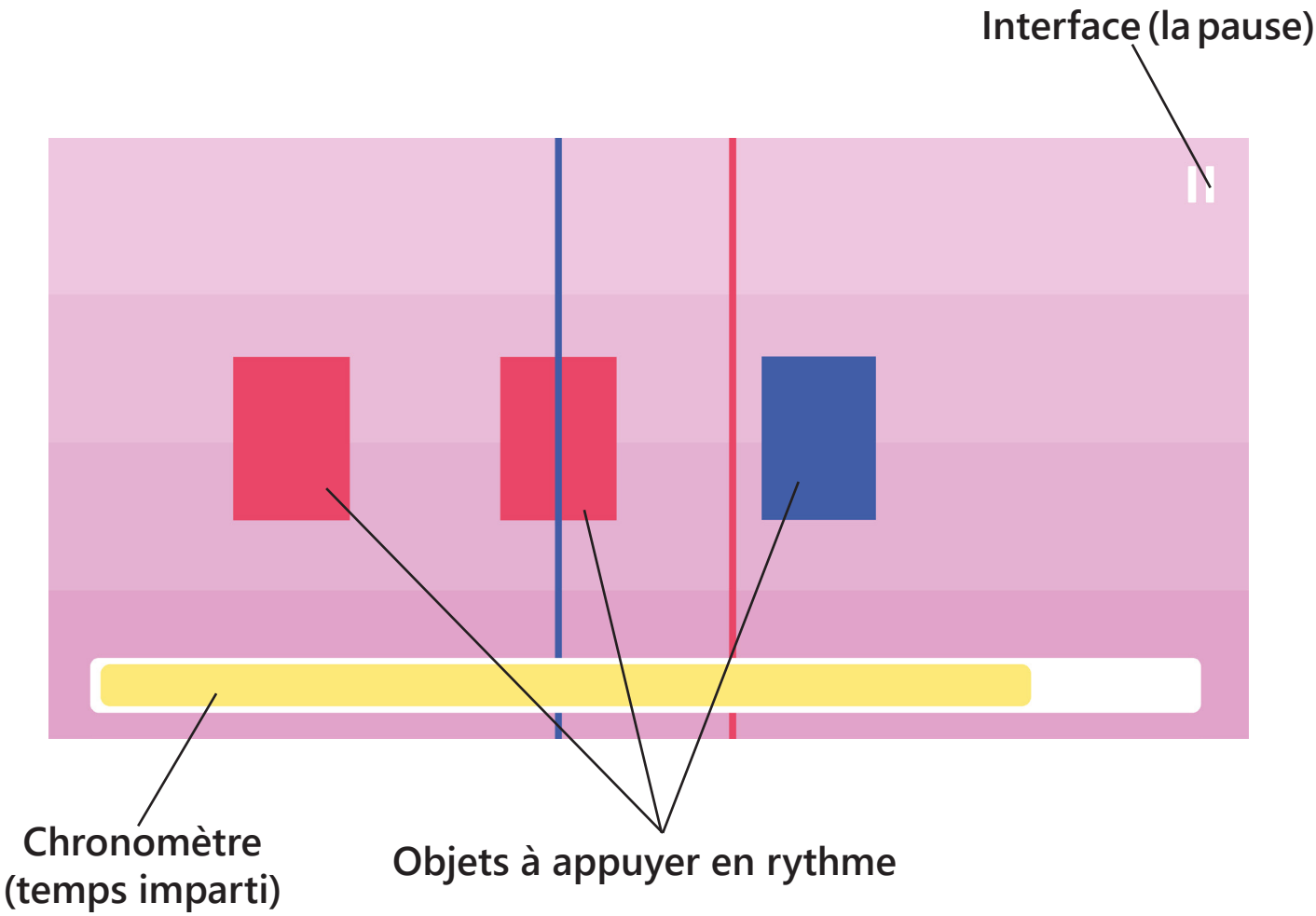


Puis sur la nôte de droite, au bon moment



Et ainsi de suite

Maquette


Interface (la pause)
Chronomètre
(temps imparti)
Objets à appuyer en rythme