

Four pixel art clouds are scattered across the top of the image. Each cloud is composed of a simple black outline and a light gray fill, giving them a retro, blocky appearance.

JSRASSIK

[*http://jsrassik.xyz*](http://jsrassik.xyz)





I. CONTEXTE

Enfant, j'aimais les dinosaures (comme beaucoup d'enfants),
d'ailleurs qui n'aime pas les dinosaures ?

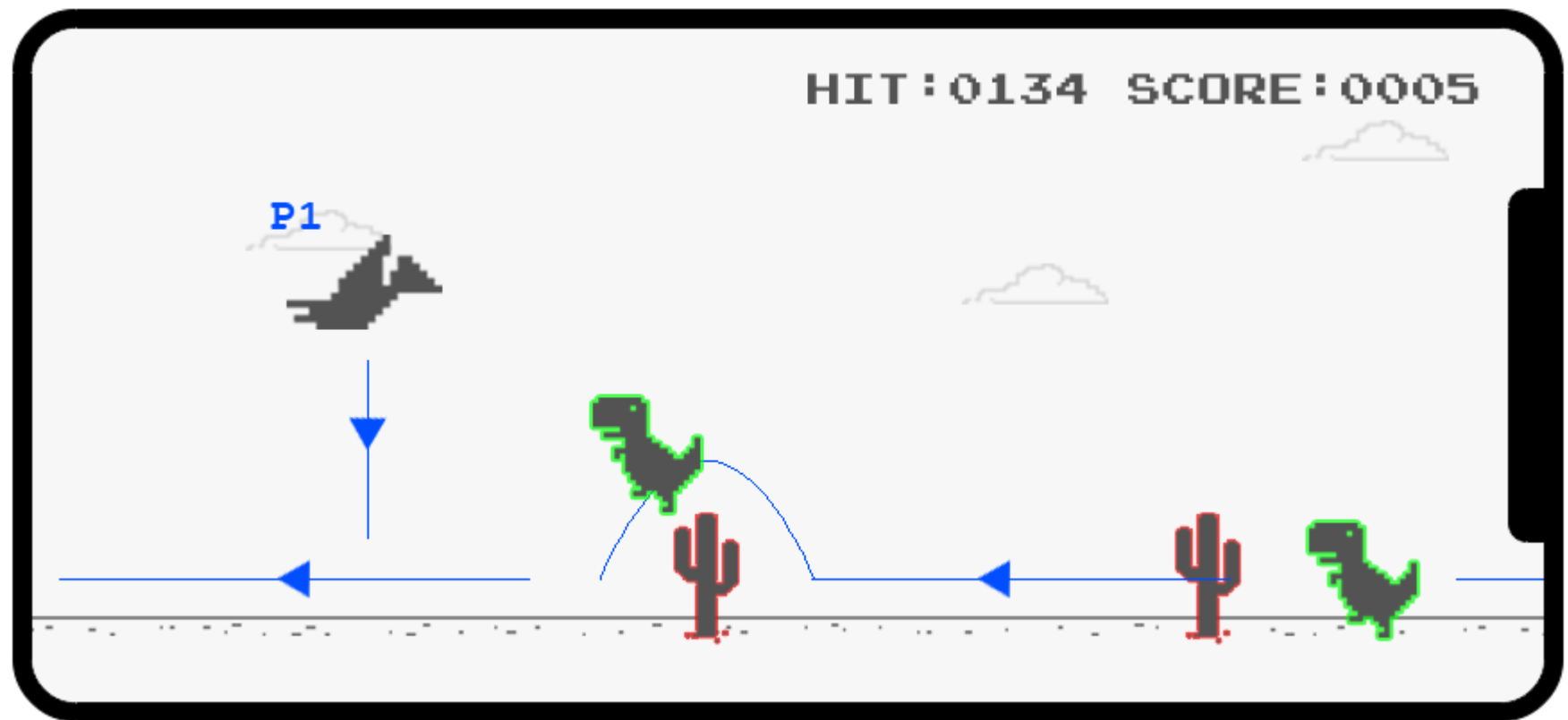


En attendant les progrès en clonage d'animaux préhistoriques,
je propose un petit jeu dont le but est d'incarner un
ptérodactyle qui chasse des tyrannosaures.

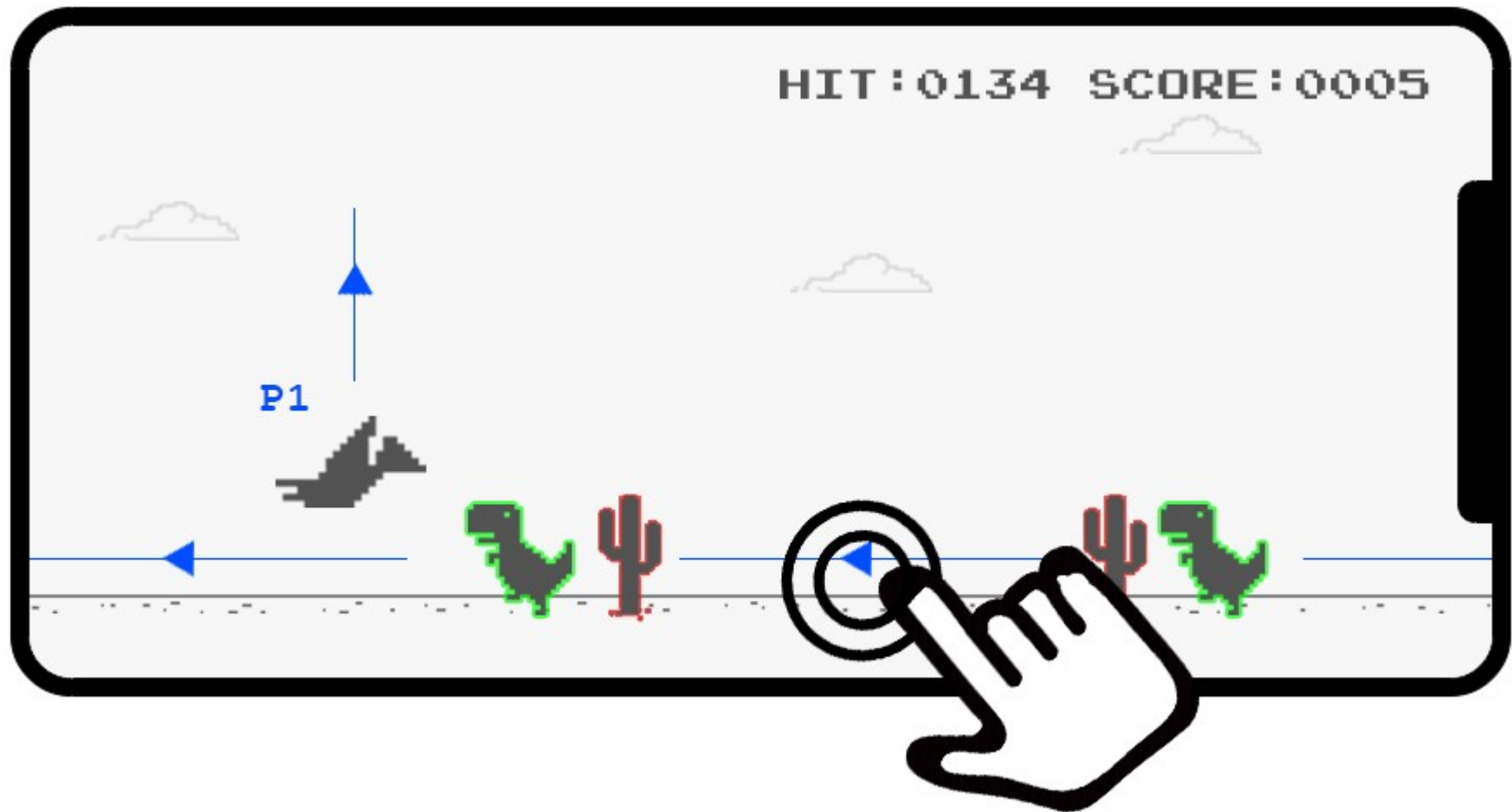
Je me suis inspiré des mécaniques de Flappy Bird  et des
textures ainsi que de certaines mécaniques de Google T-rex-
runner  pour imaginer ce jeu.



II. MAQUETTES FONCTIONNELLES



II. MAQUETTES FONCTIONNELLES



II. MAQUETTES FONCTIONNELLES



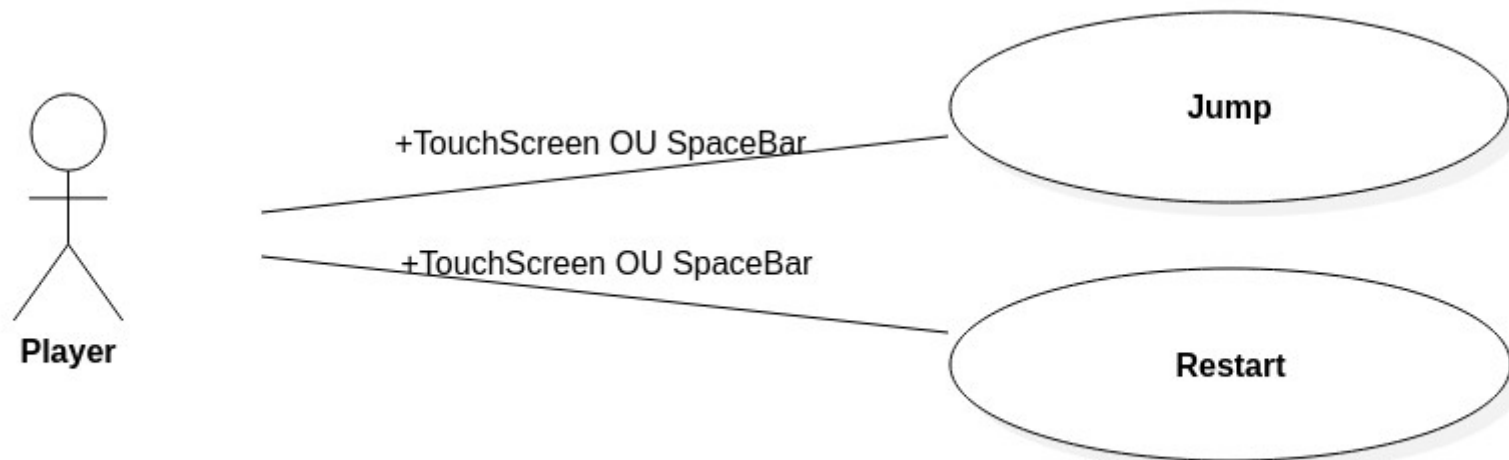
III.COMMANDES EN JEU & DIAGRAMMES D'UTILISATION

Pression n'importe où sur l'écran (si ce dernier est tactile).

- Via l'évènement `touchstart`.

Pression sur la barre d'espace.

- Via l'évènement `keydown`.



IV. EXPLICATION DE L'ARCHITECTURE DU PROJET

Le jeu est basé sur un <canvas>.

La fonction `startInterval()` va exécuter toutes les 14 millisecondes la fonction `intervalLoop()` qui elle même va exécuter toutes les fonctions utiles au rafraîchissement du canvas, comme le déplacement des décors ou des dinosaures.

```
startInterval() {  
  if (!this.intervalStarted) {  
    this.intervalStarted = true;  
    this.interval = setInterval(() => { this.intervalLoop() },  
  }  
}
```



IV. EXPLICATION DE L'ARCHITECTURE DU PROJET

Les collisions ont été la partie la plus compliquée à programmer.

Un tableau contenant les informations de collisions est créé au lancement du jeu, puis celui-ci est modifié toutes les 14ms.

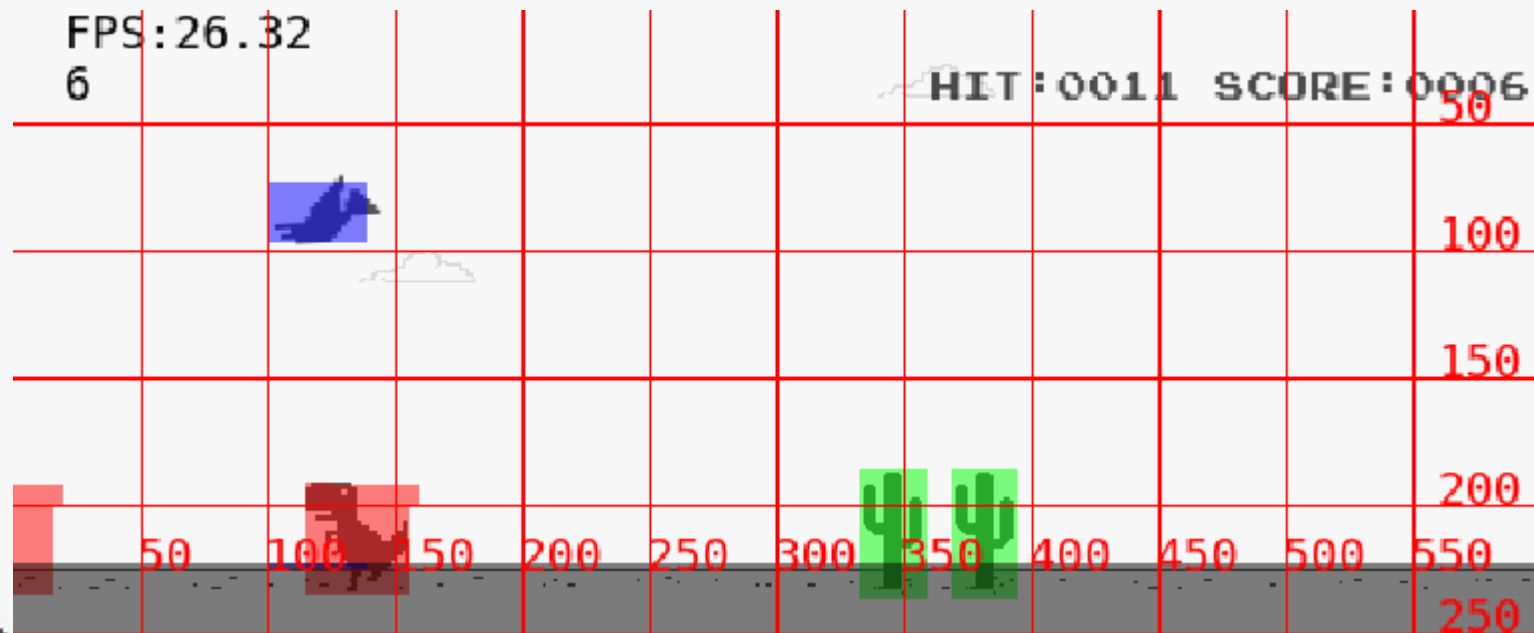
Chaque pixel du canvas correspond à une variable de collision du tableau.

```
// Initialisation du système de collision.
initCollision() {
  for (let y = 0; y < this.parent.HEIGHT; y++) {
    this.parentcollisionArray[y] = [];
    for (let x = 0; x < this.parent.WIDTH+100; x++) {
      this.parentcollisionArray[y][x] = { y: y, x: x,
      player: 0, ground: 0, cactus: 0, leaf: 0, trex: 0 };
    }
  }
}
```


IV. EXPLICATION DE L'ARCHITECTURE DU PROJET

Toutes les 14 secondes, les instances des objets des tyrannosaures vont vérifier s'ils sont en collision avec le sol, avec un cactus ou avec le joueur, même chose pour l'instance de l'objet du joueur (ptérodactyle).

Le jeu avec les collision affichées :



V. ORGANISATION DU PROJET

Le projet à été organisé en 4 sprint d'une semaine :

- [Semaine 1 \(5 août 2019 – 9 août 2019\)](#) : Maquettage du projet, organisation du début du projet, création des bases du jeu.
 - [Semaine 2 \(11 août 2019 – 14 août 2019\)](#) : Réorganisation du projet en P00, création du système de collision.
 - [Semaine 3 \(26 août 2019 – 30 août 2019\)](#) : Deuxième réorganisation du projet en P00, ajout du ptérodactyle et de ses contrôles.
 - [Semaine 4 \(2 septembre 2019 – 6 septembre 2019\)](#) : Adaptation du jeu suivant le type d'appareils (téléphone mobile ou ordinateur de bureau) et ajout du message de fin de jeu.
-
- Le projet est constitué de 14 fichiers Javascript dont 986 lignes de code (commentaires et sauts de lignes expurgés).





A VOUS DE JOUER SUR

[*http://jsrassik.xyz*](http://jsrassik.xyz) !

(Mon meilleur score est 43 !)

