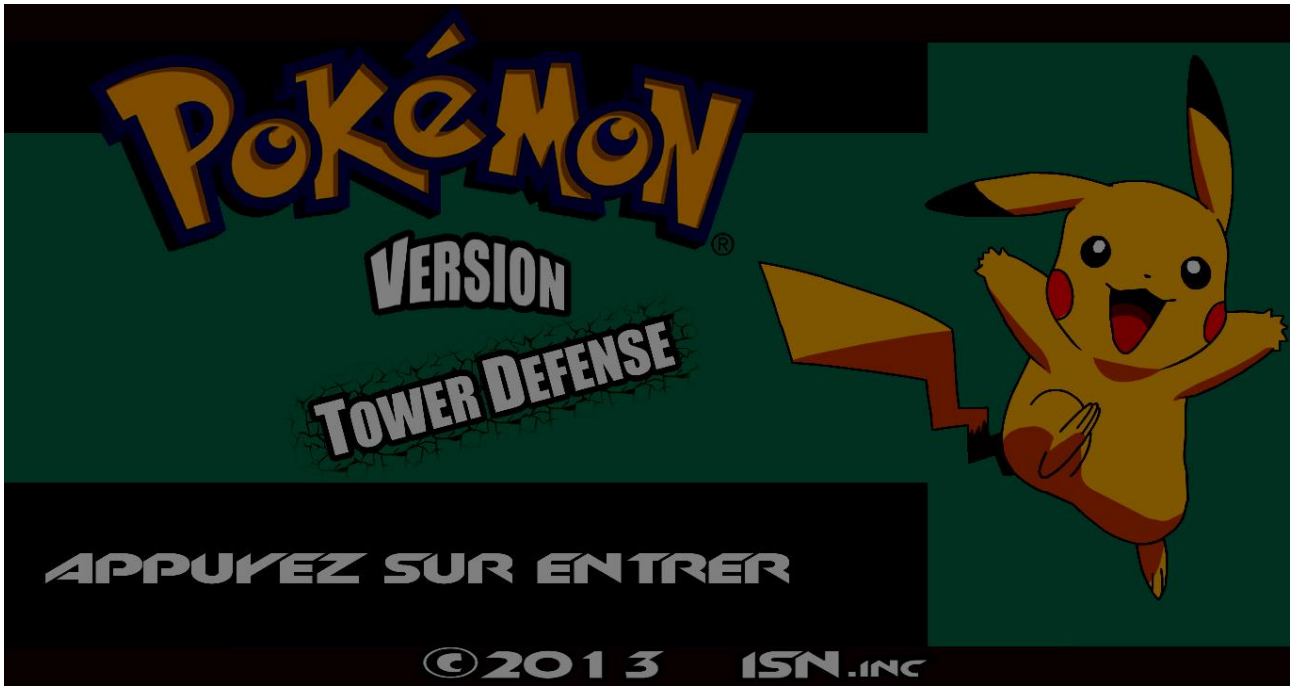
Jeux multi-joueurs, projet ISN

Par Matthias Creach
Jonas Bouscharain
Clément Lamoureux
Elèves de TS2
Lycée Aristide Briand



http://isn.codelab.info/e-portfolio/e-portfolios-du-groupe-vendredi/clement-ts2/mes-mini-projets-isn/mon-cv/

Table des matières

Problématique :

Comment créer un jeu et le mettre sur une plateforme en ligne pour le rendre accessible au multi-joueurs ?

Pourquoi ?

Ce projet nous a bien intéressés car, étant fan de jeux vidéo, nous avons toujours voulu savoir comment se créait un jeu depuis la première ligne de code jusqu'à sa mise en réseaux, de plus, il ne semblait pas trop compliquer par rapport à certains autres projets, de plus, le marché des jeux vidéo avec la génération actuelle est en plein boom, ainsi, nous avons pu nous mettre dans la peau de vrais développeur et créateurs de jeux vidéo.

Positionnement du projet par rapport aux solutions existantes :

Pour mener à bien notre projet, nous avons utilisé le logiciel de codage processing qui nous a offert une banque de données (grâce notamment à son site internet) quasi inépuisable pour créer notre jeux, néanmoins, nous avons dû utiliser plusieurs autres logiciels comme Advance Map, photoshop et photophiltre

But :

Créer un jeu vidéo en ligne du type Tower defense et donc étudier comment coder un jeu et comment utiliser un réseau pour jouer en ligne.

Cahier des charges :

Objectifs :

- Création d'une carte
- Codage des attaquants
- Codage du trajet de la carte
- Réalisation de l'interface graphique du menu
- Réalisation de l'interface graphique lors des combats
- Fonction pour placer les défenseurs
- Récupération des images utilisées pour les défenseurs et les attaquants
- Ambiance sonore
- Codage de l'attaque des défenseurs
- Difficulté du jeu
- Mise en réseau

Environnement de travail utilisé et sources d'information :

Nous programmons sur processing en langage java sur des ordinateurs équipés de Windows XP et Seven. Nous avons recherché des informations sur internet et plus précisément sur le site internet de processing, dans leur base de donné (http://www.processing.org/reference/). On s'appuie pour la réalisation de ce jeu sur plusieurs exemples de tower defense, qu'ils soient sur ordinateur ou sur téléphone, on essaye de prendre les bonnes idées de chacun des jeux, on améliore ce qui ne nous parait pas coller avec notre futur jeu.

Jonas Bouscharain	Clément Lamoureux	Matthias Créach
Interface des menus	Création d'une carte et de son trajet	Fonction pour les attaquants
Interface des écrans de combat	Transparence des images Pokémons	Fonction pour placer les défenseurs
Ambiance sonore	Difficulté du jeu	Fonction attaques des défenseurs
Mise en réseau		

Qu’est-ce qu’un Tower defense ?

Un Tower defense est un jeu dans lequel deux camps s’opposent, un attaquant doit détruire une base tandis que le défenseur doit la défendre. Pour ce faire, l’attaquant envoi des monstres qui parcourront un chemin prédéfini, bien sûr, il ne pourra pas envoyer autant de monstre qu’il le veut, il y a une limite, ce qu’on appelle une vague est un nombre prédéfini lui aussi d’ennemi qu’un joueur peut envoyer en un tour. Pendant que l’attaquant enverra ses monstres, le défenseur lui devra construire des tours dans le but de détruire les monstres envoyés par l’attaquant pour défendre son château.

Création de la carte :

En premier lieu, mon travail consistait à créer une carte qu’on allait utiliser dans notre jeu, en effet, j’ai utilisé le logiciel Advanced carte pour pouvoir créer, grâce à des éléments de l’univers du jeu à partir duquel on a pioché les décors, une carte (j’ai dû placer des petits carrés de quelques pixels sur un fond d’herbes créant ainsi les éléments décoratifs de la carte) (cf*. document 1)*.

Création du trajet :

J'ai ensuite codé grâce à processing la fonction qui permet aux attaquants d'avancer sur la carte, car si ils n'avancent pas, ils ne peuvent pas espérer atteindre la fin du niveau et ainsi détruire le château, le joueur qui attaque ne pourrait donc jamais espérer gagner. Ce code fût assez simple à réaliser, en effet, il s'agit simplement de

définir des conditions qui, s'ils sont acceptés, font aller le personnage attaquants dans la direction de cette condition, par exemple, si je mets la condition gauche en marche, mon personnage ira à gauche sur la carte, sachant cela, il m'a suffi de définir des points précis de la carte ou une certaine condition deviendrait bonne, tandis que celle qui était bonne avant deviendrait fausse. Ainsi, reprenant mon exemple précédent, si je vais à gauche et que d'un coup, je veux aller en haut, je dois annuler la condition qui fait que mon personnage va à gauche, et enclencher celle qui fait que mon personnage va à droite (cf*. document 2, 3 et 5*).

Définition du placement des tours :

J'ai dû ensuite m'occuper des tours, qui défendraient le château contre les attaquants. Pour cela, j'ai dû reprendre ma carte, l'ouvrir avec un logiciel de traitement d'image et noter chacun des pixels en haut à gauche des carrés ou placer les défenseurs, car effectivement, un défenseur ne peut pas se placer n'importe où, il fallait dans la carte définir certains endroits où les défenseurs se placerait et seulement à ces endroits la, pour se faire, il fallut donc relever l'emplacement de ces zones. Ce travail la fut le plus long sans doute, car c'était un travail de précision et sur plutôt un grand nombre de carré (cf*. document 4*).

Traitement des images :

Après cela, j'ai dû récolter sur internet les images que nous allions utiliser pour notre jeu, en effet, si internet propose une banque de données d'image infinie, nous voulions rester dans notre thème en prenant des images tirées du jeu de base sur lequel nous nous basions, pour cela nous avons dû nous concerter sur quelles images allions nous

prendre, combien d'attaquants différents, combien de défenseurs, quels seraient leurs capacités spéciales etc. Une fois que nous avons pu nous mettre d'accord sur les 8 images que nous allions utiliser, il me fallait les trouver sur internet, et enlever leur fond, en effet si on ne passait pas par cette étape, il y aurait eu derrière chaque image un fond blanc, cachant ainsi la carte autour de l'image, que celle si soit en attaque ou en défense. J'ai donc pris toutes les images une par une et sur un logiciel de traitement d'image, je les ai découpées en laissant un pixel d'écart, j'ai ensuite du coller cette image sur un nouveau document pour pouvoir la redimensionner, car sans cela, elle aurait été toute petite par rapport au reste de la carte. J'ai donc redimensionné ces images en j'ai appliqué une transparence. Cette étape, moins prenante et longue que celle consistant à définir la zone de placement des défenseurs était néanmoins relativement longue, car ne serait-ce que pour un ennemis, selon son orientation et la direction qu'il prend, ce ne sont pas les mêmes images, en effet, nous voulions que notre jeu puisse être le plus vivant possible, il devait donc y avoir pour chaque direction deux images, l'un avec le pied gauche en premier, l'autre avec le pied droit, ainsi, cela donnait l'impression que l'image bougeait toute seule, ce qu'on appelle un GIF, une image animés (c'est d'ailleurs le format choisit pour chacune des images de notre jeu), on a donc pour nos quatre attaquants, quatre directions différentes, donc trente-deux images en tout à découper et auxquelles il fallait enlever le fond . Pour les défenseurs, ce fût moins long car les images étant statiques, on pouvait ne pas les faire s'animer, mais il y avait quand même 16 images à transformer. Une fois les images en notre possession, il fallait que je leur trouve des caractéristiques uniques pour chacune, en effet, nous voulions que notre jeu soit le plus complet possible même s'il n'était pas fini, c'est pourquoi j'ai dû trouver pour chaque image une capacité qui lui serait unique.

Difficulté :

Une fois cela en place, je me suis penché sur la difficulté du jeu, mais cette partie fût compliquée a réalisée car j'avais besoin du travail que mes camarades n'avaient pas encore fini, j'ai donc fait une ébauche de tout ce qu'il faudrait regarder quand on règlerait la difficulté du jeu. En effet, pour que ça ne soit pas trop facile pour l'un ou l'autre camp, il fallait régler certains détails par exemple, le coût des unités. En effet, chaque défenseur ou attaquant aura un certain coût de construction, les deux camps partiront avec une somme de départ et pourront en regagner par la suite, le défenseur en tuant des attaquants, et l'attaquant en regagnera une certaine somme par fin de niveau. Mais le coût de base de chaque unités ne devait pas être trop élevé pour aucun des deux camps, sinon, ils ne pourraient pas envoyer beaucoup voire même pas du tout d'unités, mais il ne devait pas être trop faible non plus, car si nous voulions que notre jeu ne soit pas trop dur, nous ne voulions pas non plus qu'il soit trop facile, c'est pourquoi il fallait trouver un juste milieu. Ensuite, second exemple, la vie des unités et la puissance des défenseurs, en effet si les attaquants n'ont pas le temps de faire un pas qu'ils sont déjà mort, ça n'a aucun intérêt, de plus, si un seul attaquant arrive à la base du défenseur et qu'il perd l'amusement n'est pas la non plus. Il fallait ensuite définir précisément un nombre fixe maximum d'attaquant à envoyer, car s'il pouvait en envoyer des milliers, le défenseur ne pourrait rien faire. Enfin, la vitesse d'attaque des défenseurs et la vitesse de déplacement des attaquants, car s'ils peuvent parcourir la carte en quelques secondes, ce n'est pas drôle. Autant d'exemples que de problèmes auxquels je me suis confronté.

Mise en réseau :

Je n'ai pas, pour ma part, pu m'essayer à la mise en réseau car j'étais pris par ma part de travail, néanmoins, je suivais leur avancement et ils m'expliquaient ce qu'ils trouvaient, je peux ainsi dire que si il est en réseau un jour, cela ne sera pas par l'adresse IP de l'ordinateur lui-même, en effet on a remarqué que ça n'était pas assez rapide, il faudra donc sans doute passer par une plateforme qui connecte plusieurs ordinateurs ensemble sous un même réseau.

Mise en relation de mon travail avec celui de mes camarades :

Ma carte a beaucoup servie au sein de notre équipe, en effet on a pu effectuer des tests dessus, notamment pour l'interface graphique *(cf. document 6)*, car il n'y aurait pas que ma carte affichée à l'écran, mais aussi toute l'interface graphique faite par Jonas, il a donc fallu que je modifie mon code de déplacement pour qu'il corresponde au déplacement de l'écran complet. Ensuite, toujours dans une optique graphique, il a fallu que je donne mes descriptions d'attaquants et de défenseurs à Jonas pour qu'il l'intègre à l'interface graphique et fasse en sorte que quand on place notre souris sur une image, la description correspondante s'affiche avec ses capacités spéciales etc. Ensuite, j'ai donné mon travail qui consistait à relever les pixels correspondant aux zones dans lesquelles on pouvait placer un défenseur à Matthias pour qu'il fasse toutes les zones, il fallait en effet, maintenant que j'avais les pixels, les définir dans le code. On s'est servi aussi de mes images dans

les tests, autant pour les tests d'attaques que Matthias effectuait pour tester son code avant de l'intégrer au vrai, que Jonas dans son interface ou il a dû les utiliser pour définir les boutons sur lesquels on cliquera pour lancer un attaquant ou placer un défenseur. Matthias possédait sa propre carte de test sur laquelle il test chaque ligne de code avant de l'intégrer au code, c'est aussi sur cette carte que j'ai testé le code permettant de déplacer les attaquants, ainsi, j'ai seulement eu à le retranscrire sur ma vraie carte.

Améliorations possible de notre projet :

Pour pouvoir développer notre projet, il nous faudrait du temps, je pense que c'est la seule ressource dont nous manquons actuellement, en effet, nous n'avons rien besoin d'autre que de temps pour finir ce projet, néanmoins, nous nous sommes déjà penché sur les améliorations que nous pourrions apporter à ce jeu, en outre le fait de rajouter la musique, nous pourrions intégrer le fait de pouvoir améliorer les attaquants et les défenseurs, de pouvoir choisir une voie dans laquelle se spécialiser comme par exemple, ne plus envoyer que des ennemis avec beaucoup de points de vie mais lent. L'intégration d'une intelligence artificielle pour pouvoir aussi jouer en solo, et ne plus être obliger d'avoir un équipier avec qui jouer. L'ajout d'un bouton revendre pour les défenseurs, car pour le moment, si on place un mauvais défenseur au mauvais endroit, on ne peut plus le déplacer. On pourrait aussi essayer de mettre en place plusieurs modes de difficultés. Par contre, je ne pense pas que l'interface ai grandement besoin d'être améliorée, en ayant travaillé tous les trois à l'améliorée, elle est très bien pour un jeu comme le nôtre.

Expérience personnelle :

Personnellement, ce travail m'a apporté beaucoup, tant que le plan informatique car j'ai appris beaucoup de choses concernant le code et l'utilisation de logiciel de retouche graphique grâce à mes camarades et à internet, que sur le plan partenarial, en effet, de travailler au sein d'une équipe, même si ce n'est pas la première fois est toujours quelque chose d'enrichissant, car entre le fait qu'il faille rendre le travail à l'heure, et du travail bien fait, on est servi. De plus, je trouve ça plus facile de travailler avec des personnes que l'on côtoie tous les jours, car si on arrive pas quelque chose, on a moins peur de leur demander, de plus, on essaye de faire le maximum pour leur plaire, car on ne veut pas les décevoir au risque de travailler par la suite dans une ambiance bien moins bonne que les première séances. C'est pourquoi je dis que c'est une expérience très enrichissantes, cela nous apprends les bases du travail, et ça nous sera utile quand plus tard, nous devrons travailler en groupes autour d'un même projet.

Lien vers Github :

Dossier : https://github.com/codelab-info/boulam/tree/master/Dossiers

Diaporama : https://github.com/codelab-info/boulam/tree/master/Diapos

Annexes

Document 1 :

Document 2 :

```
void trajetMap1()
{
  if(x > 0 && y == 460)//puisque les pixels x et y de l'attaquant remplissent les conditions,
  {
    gauche = true;//l'attaquant va à gauche
  }
  if(x == 50 && y == 460)// une fois le pixel en haut a gauche de l'attaquant arrivé au point x = 50 et y = 450
  {
    gauche = false;//l'attaquant arrête d'aller à gauche
    haut = true;//l'attaquant va en haut
  }
  if(y == 270 && x == 50)// une fois le pixel en haut a gauche de l'attaquant arrivé au point x = 50 et y = 270
  {
    haut = false;//l'attaquant arrête d'aller en haut
    droite = true;//l'attaquant va à droite
  }
  if(x == 580 && y == 270)// une fois le pixel en haut a gauche de l'attaquant arrivé au point x = 580 et y = 270
  {
    droite = false;//l'attaquant arrête d'aller à droite
    bas = true;//l'attaquant va en bas
  }
  if( y == 355 && x == 580 )// une fois le pixel en haut a gauche de l'attaquant arrivé au point x = 580 et y = 355
  {
    bas = false;//l'attaquant arrête d'aller en bas
    gauche = true;//l'attaquant va à gauche
  }
  if( x == 320 && y == 355 )// une fois le pixel en haut a gauche de l'attaquant arrivé au point x = 320 et y = 355
  {
   gauche = false;//l'attaquant arrête d'aller à gauche
    haut = true;//l'attaquant va en haut
  }
  if ( x==320 && y == 50)// une fois le pixel en haut a gauche de l'attaquant arrivé au point x = 320 et y = 45
  { haut = false;//l'attaquant arrête d'aller en haut, il disparait donc de la carte
  }
}
```

Document 3 :

Document 4 :

(Les croix créées par l'intersection des deux traits noirs est le pixel que j'ai dû relever)

Document 5 :

```
PImage Map1;

PImage ChenipanL, ChenipanL2;//on définit les images qui vont nous servir à faire bouger l'ennemi
PImage ChenipanR, ChenipanR2;
PImage ChenipanD, ChenipanD2;
PImage ChenipanU, ChenipanU2;

int x, n;
int x2, y2;
float y;
boolean chen, gauche, droite, bas, haut;//on définit les boléans qui serviront a donner une direction a l'ennemi
boolean poser;// ce boléan sert pour le placement des tours

void setup()
{
  Map1 = loadImage("Map1.png");

  ChenipanL = loadImage("ChenipanL.gif");//on doit charger toutes les images, deux pour chaquemouvement
  ChenipanL2 = loadImage("ChenipanL2.gif");//pour donner l'impression que l'ennemi est animer
  ChenipanR = loadImage("ChenipanR.gif");//et marche
  ChenipanR2 = loadImage("ChenipanR2.gif");
  ChenipanD = loadImage("ChenipanD.gif");
  ChenipanD2 = loadImage("ChenipanD2.gif");
  ChenipanU = loadImage("ChenipanU.gif");
  ChenipanU2 = loadImage ("ChenipanU2.gif");

  size(Map1.width, Map1.height);//on adapte la taille de la carte directement a la taille de l'écran
```

```
  chen = true;//on commence par mettre le boléan chen en true, cela servira plus tard à pouvoir
  //lancer plusieurs ennemis différents, en activant un tel ou un autre boléan
  x = 690;//on fait commencer son pixel en haut a gauche, en x = 690
  y = 460;// et en y = 460
}

void draw()
{
  image(Mapl, 0, 0);
  noFill();
  {
    fill(0, 255, 255);
    {
    }
  }

  mouvement();//procédures qui seront réutilisée plus tard
  trajetMapl();

  if(gauche)// si mon ennemi va a gauche
  {
    if(chen)// si l'ennemi sélectionné est celui la ( ceci est dans une optique d'évolution )
    {
      n++;// alors la variable n s'incrémente
      image(ChenipanL, x, y);// et l'image ChenipanL (la première des deux) s'affiche
      if(n == 20)// mais si on arrive a n=20 (valeur déterminée après de nombreux essaie)
      {
        chen = false;// alors la boléan chen devient false
      }
    }
    else
    {
      n--;////ensuite, le n se décrémente
      image(ChenipanL2, x, y);// et jusqu'a ce qu'il arrive a 0, c'est la deuxième image qui est affichée
      if(n == 0)//enfin, quand le n redevient 0,
      {
        chen = true;// le boléan redevient true.
      }
    }
  }
  if(droite)
  {
    if(chen)
    {
      n++;
      image(ChenipanR, x, y);
      if(n == 20)
      {
        chen = false;
      }
    }
    else
    {
      n--;
      image(ChenipanR2, x, y);
      if(n == 0)
```

```
      {
        chen = true;
      }
    }
  }
  if(bas)
  {
    if(chen)
    {
      n++;
      image(ChenipanD, x, y);
      if(n == 20)
      {
        chen = false;
      }
    }
    else
    {
      n--;
      image(ChenipanD2, x, y);
      if(n == 0)
      {
        chen = true;
      }
    }
  }
     if(haut)
  {
```

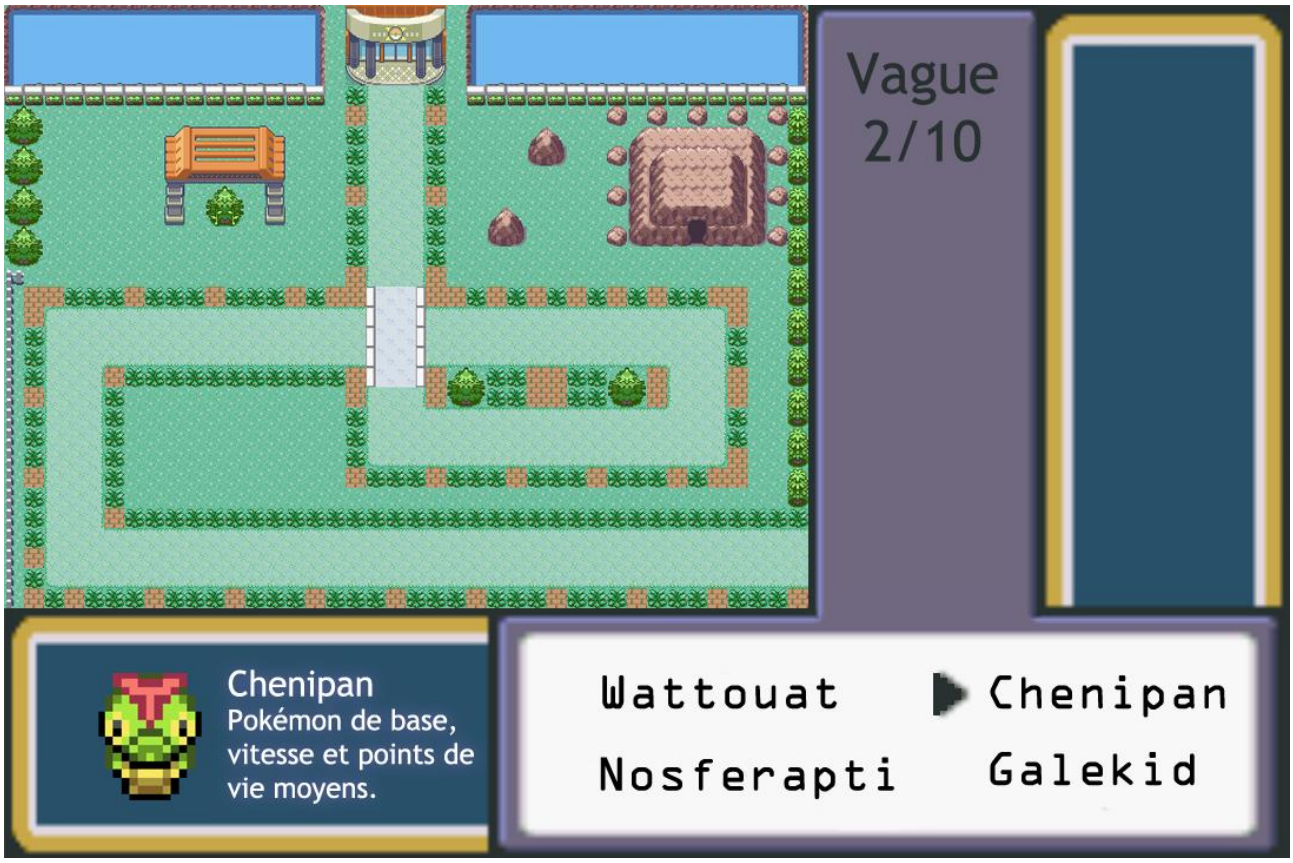
```
    if(chen)
    {
      n++;
      image(ChenipanU, x, y);
      if(n == 20)
      {
        chen = false;
      }
    }
    else
    {
      n--;
      image(ChenipanU2, x, y);
      if(n == 0)
      {
        chen = true;
      }
    }
  }
}

void mouvement()//cette procédure fait écho à la void trajet Mapl, en effet,
//les droite, gauche, haut et bas seront réutilisées ici
//en effet, nos boléans haut, bas, droite et gauche du début sont ici définies
{
  if(droite)//pour aller a droite
  {
    x++;// le x doit s'incrémenter ( sur un axe (o, i; j), plus on va a droite, plus le x est grand
```

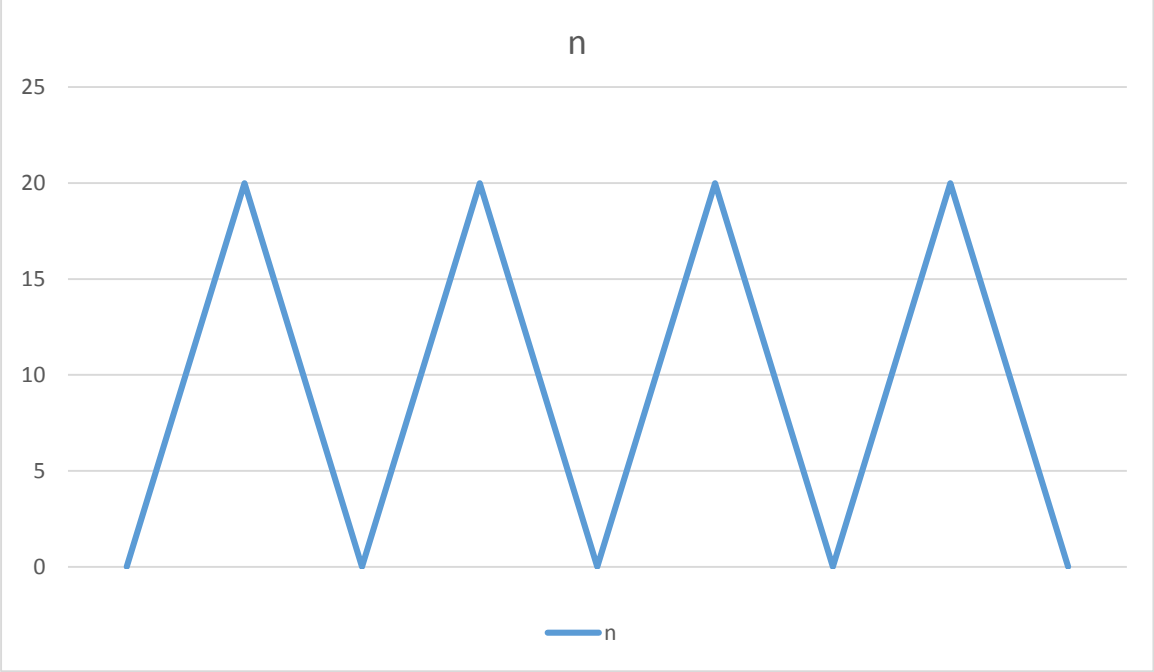
```
    }
    if(gauche)//donc pour aller a gauche
    {
      x--;//le x doit décrémenter
    }
    if(bas)//pour aller en bas ( toujours sur l'axe (o, i; j), mais cette fois, pour ce logiciel,
    //plus on va en bas, plus le y est grand
    {
      y++;//donc plus le y doit s'incrémenter
    }
    if(haut)//enfin, pour aller en haut
    {
      y--;//le y doit décrémenter
    }
}
```

(Pour comprendre cette histoire d'alternance des images, je vous invite à regarder le document 7)

Document 6 :



Document 7 :



A chaque fois que la courbe atteint le 20, elle prend l'image 2, quand elle retombe à 0, elle reprend l'image 1 et ainsi de suite en échangeant à chaque fois.