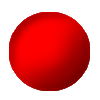


MEMBRE DU GROUPE :

*DIPOKO William*

*MAGNI Lorenzo*



SOMMAIRE

[I. Présentation du projet « Pacboy » 3](#_Toc452417703)

[II. Analyse de besoins 6](#_Toc452417704)

[III. Recherche d’idées 7](#_Toc452417705)

[IV. Répartition des tâches 8](#_Toc452417706)

[V. Réalisation 10](#_Toc452417707)

[VI. Intégration de mon travail 14](#_Toc452417708)

[VII. Bilan et perspectives 15](#_Toc452417709)

[VIII. Diffusion du projet 17](#_Toc452417710)

[IX. Annexes 18](#_Toc452417711)

# Présentation du projet « Pacboy »

Dans ce dossier nous traiterons de notre projet intitulé Pacboy. Pacboy est entièrement réalisé en langage python, et, s’inspire d’un fameux jeu du nom de « Pacman ». Bien que les noms soient semblables, les règles du jeu sont différentes.

Pacboy est un jeu à deux joueurs, nécessitant les touches (Haut / Bas / Gauche / Droite) et (Z / Q / S / D). Deux pions, les pacboys, s’affrontent, en trois manches, sur une grille (10x7) parsemée d’obstacles et de pastilles de couleurs jaunes et violettes.

Le but du jeu est d’arriver à l’extrémité de la grille en ayant "mangé" le plus de pastilles possible, car, celles-ci rapportent des points permettant de déterminer le gagnant. Cependant une autre règle s’ajoute : une manche ne peut être terminée s’il reste une pastille violette. Ainsi pour enclencher la manche suivante, les joueurs doivent obligatoirement manger toutes les pastilles violettes.

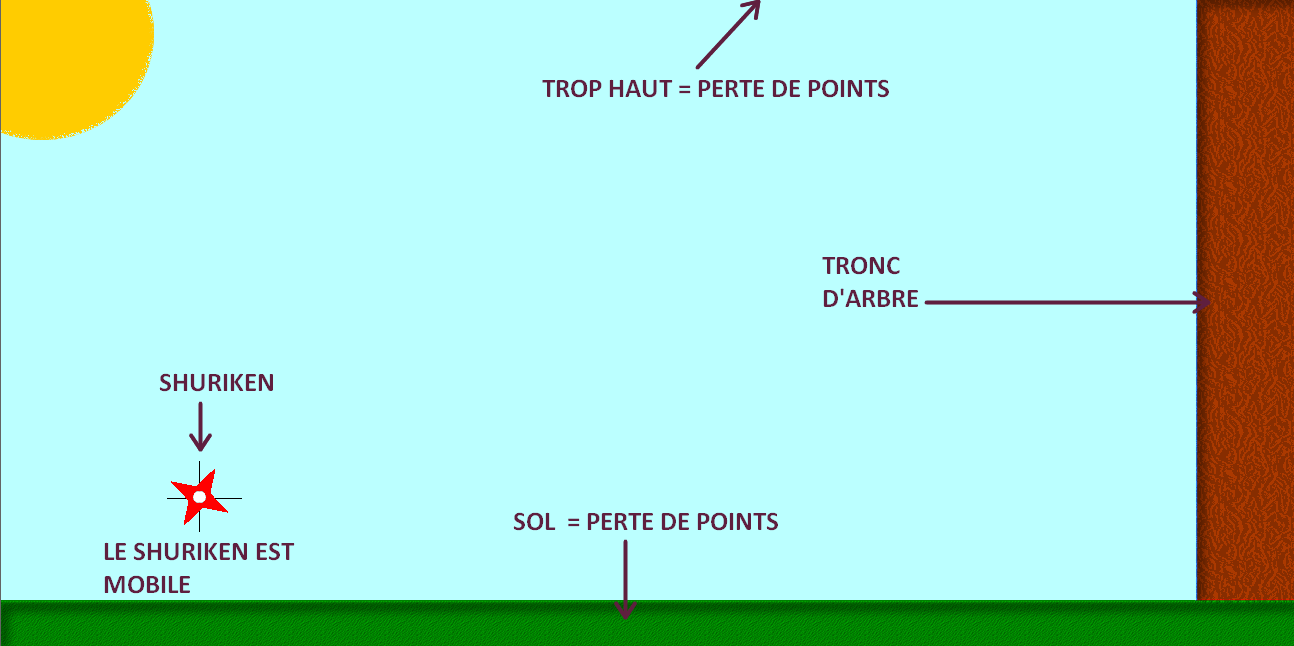
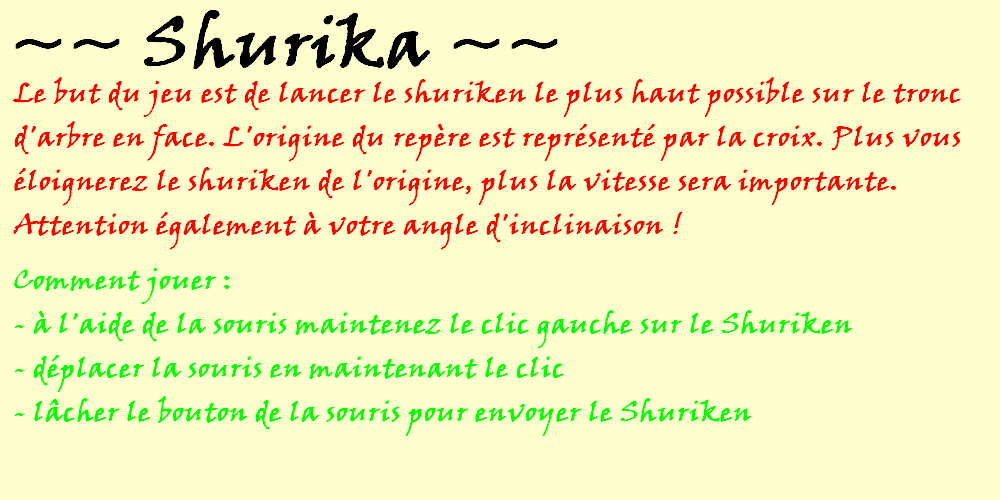
Les points acquis s’ajoutent à un compteur qui permet à chaque fin de manche de calculer le score de chaque joueur. En plus du compteur, le temps est un autre facteur influençant le score : plus le temps de la manche sera élevé plus le score sera bas.

*Bon à savoir :*

Dans le jeu Pacboy, sont intégrés deux mini-jeux : « Shurika » et « Blast ». Chaque pastille violette est liée à un de ces jeux. Le joueur qui mange la pastille est mis à l’épreuve dans un mini-jeu où ses points sont en danger. Plus il réussira dans le mini-jeu en question, plus il gagnera de points. Cependant perdre dans l’un des deux jeux équivaut à une perte de quelques points.

*La question qui se pose alors est quels sont ces mini-jeux ?*

*Shurika (De Lorenzo)*



*Bl@st (de William)*





# Analyse de besoins

Étant de grands "gamers" dans l'équipe, créer un jeu est pour nous une étape indispensable. Nous recherchions donc à en apprendre d'avantage pour programmer à notre niveau. C'est donc une étape parmi d'autres projets futurs (jeu ou non).

Ayant seulement quelques bases en programmation dans notre groupe et l’envie de progresser, le jeu, dès le départ, nous a semblé être un moyen ludique d’avancer dans ce domaine.

*Que savons-nous sur les jeux ?*

Un jeu est un programme ayant pour objectif premier d’être divertissant grâce à son aspect esthétique et ses règles qui déterminent le but du jeu, les possibilités et les interdictions. Donc au niveau du programme, un jeu serait une suite d’événements qui une fois rassemblées donnent une forme au jeu.

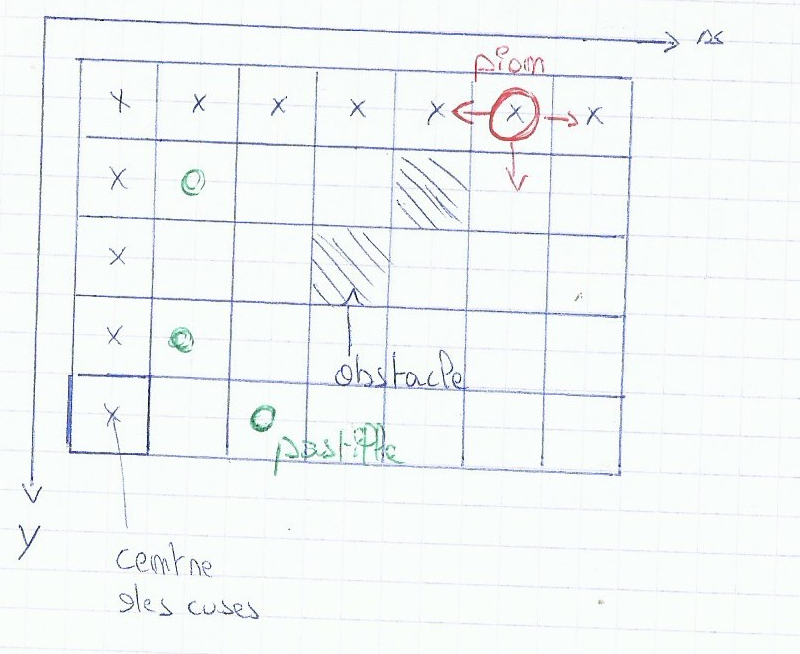
# Recherche d’idées

Notre programme s’appuie principalement sur notre expérience en jeux, surtout de mini-jeux. Nous avons repensé à ces mini-jeux auxquels nous avons joué dans le passé et encore maintenant. Nous avons puisé dans divers éléments existants et dans notre imagination pour créer un jeu à notre image.

L’une des étapes très importante pour démarrer était de formuler avec une phrase, ce que nous souhaitions faire. Ensuite cette phrase devait être traduite avec le code. Ex : Appuyer la flèche du haut fait monter le pion d’une case sur la grille. Monter une case équivaut à remonter sur l’axe des ordonnées donc à soustraire la valeur du mouvement à l’ordonnée du pion.

Nous avions donc les idées en tête, maintenant il nous a fallu les réaliser. Nous venions de commencer la programmation et nous voulions des images, du son, la possibilité de faire des calculs, avoir une interface graphique … C’est pourquoi nous avons décidé d’écrire entièrement le jeu en Python. Python est très efficace car c’est un langage complet : il existe des bibliothèques pour pratiquement toutes les opérations. Pygame (pour le son), math (pour les calculs), tkinter (pour les graphismes).

Bien entendu, leur utilisation n’a pas été maîtrisée d’un claquement de doigts. Nous avons mené des recherches dans un livre, sur les forums, sur des sites de cours afin d’aboutir à notre production (voir webographie et bibliographie).



*Schéma réalisé au cours de*

*L’année pour visualiser*

# Répartition des tâches

|  |  |  |
| --- | --- | --- |
| **Séance** | **Lorenzo (MOI)** | **William** |
| N°1 | * Création de la grille * Extrémités de la grille * Images des pions (identiques sauf la couleur) | * Chargement des images dans le programme * Affichage sur la grille |
| N°2 | * Création des obstacles : affichage aléatoire sur la grille | * Établissement de la fonction gérant le mouvement des pions sur la grille |
| N°3 | * Création des pastilles : images + affichage sur la grille | * Établissement des conditions permettant aux pions de ne pas sortir de la grille |
| N°4 | * Réarrangement de la fonction mouvement des pions en vue de gérer les collisions * Gestion des collisions entre les pions et les obstacles | * Initialisation et intégration des variables compteurs, score, manche dans le code |
| N°5 | * Gestion de la suppression des pastilles au contact d’un pion * Gestion de la fin des manches + parties | * Création des fenêtres à la fin d’une partie * Création des menus affichés * Écriture des règles |
| N°6 | * Ajout du tableau des scores | * Gestion du temps dans le code, changement du calcul de score en fonction du temps |
| N°7 | * Changement des images : pions, pastilles * Recherche + Ajout du son dans le jeu | * Recherche d’idées pour compléter le jeu |
| N°8 | * Création du décompte avec sons | * Recherches pour la création d’un mini-jeu |
| N°9-12 | * Création entière du jeu « Shurika » * Intégration dans le jeu Pacboy | * Création entière du jeu « Bl@st » |

A chaque fin de cours d’ISN on se fixe les objectifs : ce qui reste à faire et lequel d’entre nous doit le faire : Pendant la semaine on discute des améliorations possibles, on recherche les ressources nécessaires et on programme. Pendant le cours on se concerte, et nous rassemblons les parties de code. Nous communiquons principalement par SMS, mais aussi par mail pour s’envoyer les programmes en .py ou directement au lycée en échangeant nos clés USB. Nous avons d'ailleurs ouvert un compte dropbox pour pouvoir partager des documents aisément.

*Pour résumer :*

La totalité de notre programme nous a demandé 12 séances. Sachant qu’il fallait à chaque fois finaliser notre travail chez nous avec de la recherche. De plus chacun d’entre nous a réalisé un jeu de son côté, et, a dû l’intégrer à Pacboy.

# Réalisation

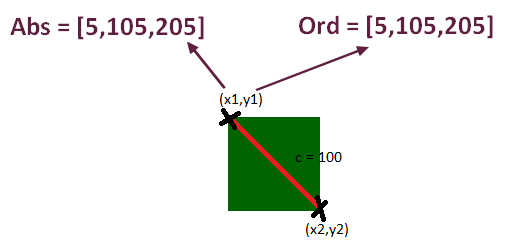
**1 – La Grille (10x7)**

C’est notre plateau de jeu. *Pourquoi 10x7 ?* La réponse peut paraître « bête » mais une grille un peu plus grande sortirait de l’écran de notre PC sachant que nous voulions aussi afficher le score à côté. Ainsi la grille est basée sur la création de lignes séparées de 100 pixels. Les tracés des lignes sont en fonction d’une suite de la forme : 100\*i + 5.

Le problème que j’ai rencontré ici est le fait qu’au départ, il s’agissait d’une suite géométrique de la forme (100\*i), or, avec cette dernière les premiers tracés de la grille étaient confondu avec les bords de notre fenêtre, donc ils n’apparaissaient pas. D’où le «  + 5 » qui permet de les décaler de 5px.

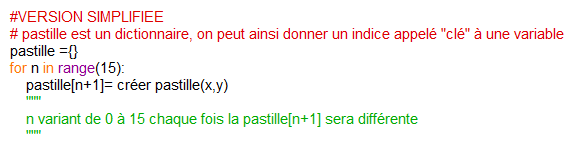
**2 – Obstacles et Pastilles**

Les obstacles sont des carrés dessinés dans l’interface à l’aide de la méthode "can.create\_rectangle(x1, y1, x2, y2)" pour les placer on choisit **aléatoirement** les valeurs x1 et y1, puis on ajoute 100 à chacune pour obtenir x2 et y2. Pour les choisir aléatoirement j’ai opté pour des Listes. En effet **chaque croisement** entre les lignes verticales et horizontales de notre grille, ont des abscisses et des ordonnée connues. Ainsi il suffit de stocker ces valeurs dans deux listes Abscisses et Ordonnées puis de choisir aléatoirement une grâce au module « random » de Python.



Les pastilles sont des images d’un cercle jaune de coordonnées (x,y) placées au **centre** des cases. De la même manière que les croisements des lignes (pour les obstacles), les coordonnées des centres de chaque case sont connues et stockées dans des listes. Donc x et y sont choisis aléatoirement dans les listes Centres\_X et Centres\_Y.

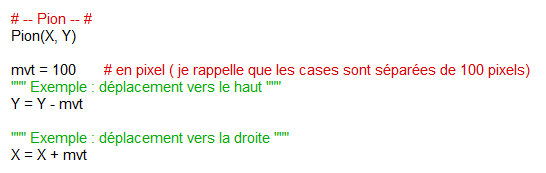
La difficulté majeure rencontrée ici fut de pouvoir **distinguer** une pastille d’une autre : pour pouvoir supprimer une pastille précisément et non pas la dernière crée lorsqu’un pion passe dessus. *Pour cela j’ai utilisé un dictionnaire qui me permet donc de donner un indice à une variable et donc de distinguer par exemple la pastille 1 de la pastille 2.* Je n’aurais jamais découvert cette astuce très utile, si je n’avais pas été confronté à ce problème.



**3 – Mouvement**

Le pion est initialisé grâce à ses coordonnées (x,y) et doit être au centre de la case, alors je m’aide des valeurs contenues dans Centres\_X et Centres\_Y pour initialiser le pion au centre d’une case.

Comme pour un simple algorithme, pour déplacer le pion, il suffit d’écraser l’ancienne valeur connue de ses coordonnées par une nouvelle valeur obtenue par l’ajout de 100 pixels à l’ancienne valeur.

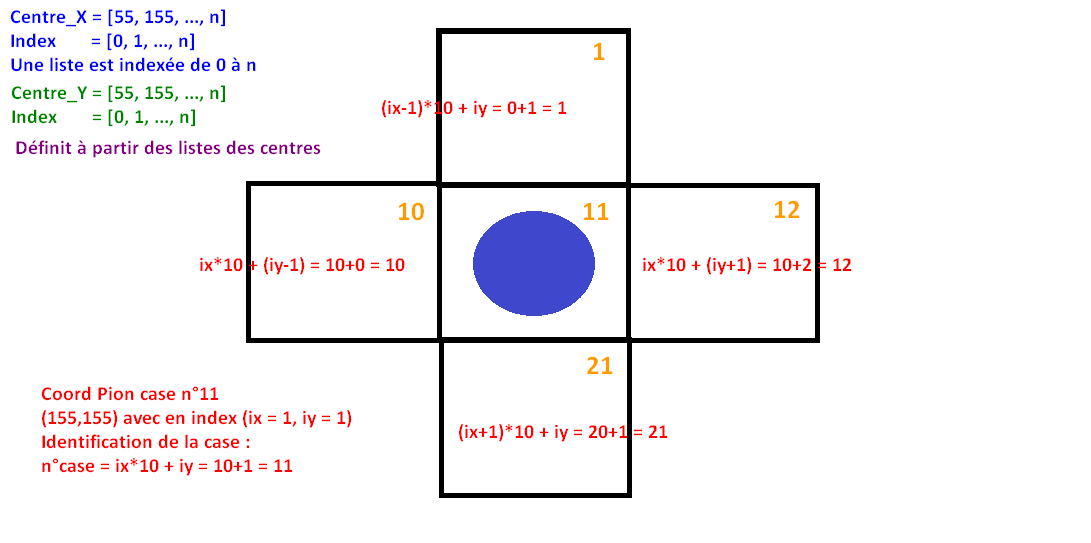
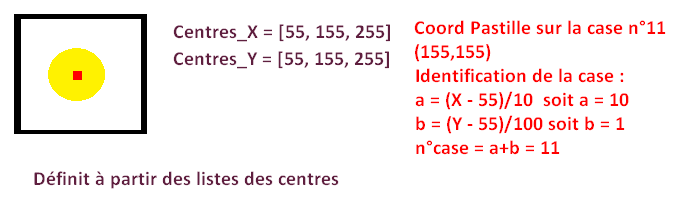
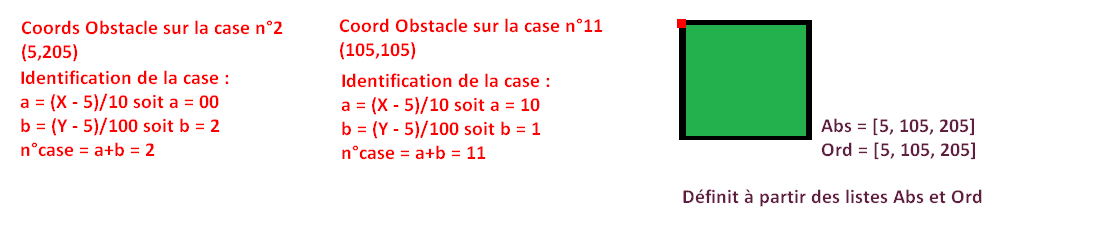


**4 – Collision et suppression des pastilles**

La principale difficulté dans cette partie fut de visualiser parfaitement à quoi correspond une case, ou plutôt comment définir une case sur la grille. J’ai pour cela fait un croquis sur une feuille. Ainsi j’ai pu établir la numérotation des cases.

Une grille 10x7 équivaut à 70 cases. J’ai décidé de les numéroter de 0 à 69. Ainsi il faut maintenant connaitre les cases des **obstacles**, des **pastilles**, des **pions** et celles **autour des pions**. La difficulté de définir une case selon les cas, a été importante mais une fois comprise, ça a été simple.

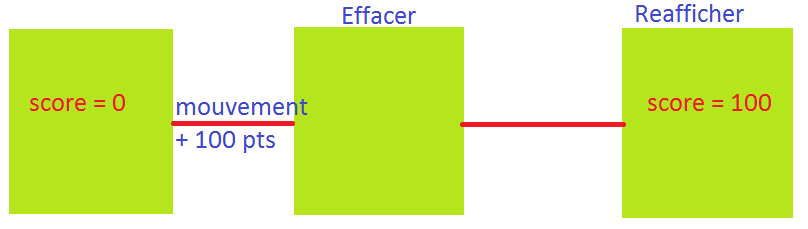
***Résumer en image : (3 méthodes possibles)***



Avec ces méthodes de reconnaissance des cases, on peut savoir à chaque mouvement, si le pion est sur la même case qu’une pastille et donc la supprimer. On peut aussi savoir si l’une des cases autour du pion contient un obstacle et donc, ne pas autoriser le déplacement.

**5 – Tableau de score**

Le tableau de score n’a pas été bien compliqué à comprendre. Il suffisait d’effacer ce qu’il contient et tout réafficher à chaque mouvement.



**6 – Jeu entier Shurika [lien en annexe]**

Créer un jeu en entier a été assez difficile, cette fois ci, je devais gérer tout, et seul. La grosse difficulté sur le jeu Shurika a été de mobiliser mes connaissances de terminale et des classes antérieures car il demandait beaucoup d’opérations : calculs d’angles (Théorème d’Al Kashi), équation horaire (mécanique Newton)….

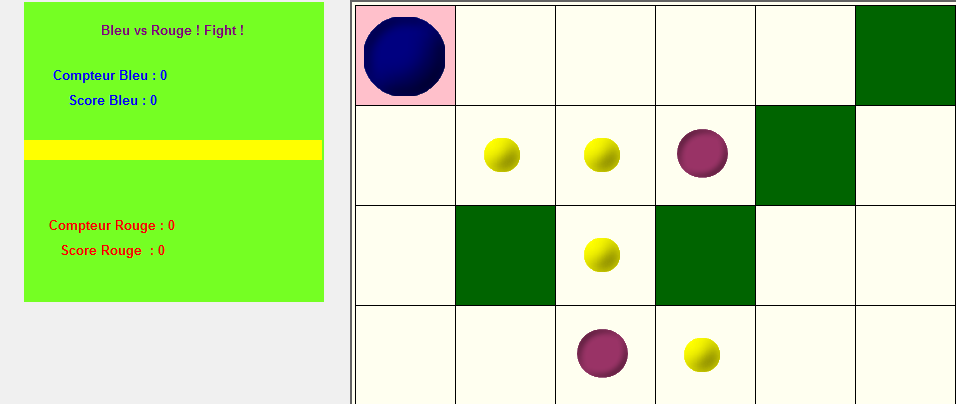
Mais ça a été une bonne expérience, car j’ai pu m’investir en autonomie. Dans notre vie professionnelle je ne serai pas toujours accompagné.

# Intégration de mon travail

De mon propre point de vue, l’investissement dans ce projet a été important. Ainsi de nombreuses fonctions contiennent une partie de mon travail. Au final en lançant le jeu, on peut voir :

Le plateau de jeu, les images, les obstacles, les pastilles, le tableau de score, le décompte

Une pastille violette lance mon jeu Shurika (cf. images plus haut)



*Par ailleurs, une grande partie des commentaires a été rédigé par moi-même.*

# Bilan et perspectives

Avec plus de temps à consacrer au projet, il serait tout à fait possible d’obtenir un jeu plus aboutit :

* Agrandir la grille et d’ajouter des obstacles/pastilles, qui seraient intéressant pour corser les parties.
* Au lieu de laisser les obstacles se mettre aléatoirement, créer mes propres plateaux de jeu avec les obstacles à l’emplacement que je souhaite : dans l’objectif de faire plus tard un outil “création de plateau”.
* Définir de nouvelles règles de jeu basées sur les pastilles.
* Un menu plus intuitif et réaliste, avec de meilleurs graphismes, d’autres sons.
* Un système de sauvegarde de niveau, de points, de victoires (en fonction d’un pseudonyme) pouvant aboutir à un classement des meilleurs joueurs.
* Une version du jeu PacBoy en réseau, c’est-à-dire avec deux ordinateurs différents et pourquoi pas avec une grande grille avec 3 pions ou plus. Ça sera l’occasion de le distribuer à plusieurs personnes pour qu’ils puissent le tester et y jouer.

**POINT DE VUE PERSONNEL :**

Sachant que plus tard je souhaiterai travailler dans le domaine de l’informatique, j’ai pu déjà me familiariser avec un langage de programmation assez puissant, et, proposant un large choix de programmes à réaliser. J’ai donc appris l’utilisation de python, comment corriger certaines erreurs. J’ai découvert des ressources sur le langage en faisant de nombreuses recherches, et j’ai approfondit certaines connaissances (algorithme, géométrie, mécanique, trigonométrie…). Travailler sur un jeu a demandé de l’imagination, de la logique mais aussi des compétences disciplinaires.

Pour ce qui est du travail de groupe, le projet d’ISN m’a été bénéfique. J’ai connu avec mes camarades, cette ambiance d’entraide et de collaboration qui, je pense, est présente dans tous métiers demandant un travail en groupe. De plus je dois avouer que, parfois aider un camarade, ou se faire aider, apporte beaucoup à l’un et à l’autre.

Pour terminer, il a fallu aussi une bonne gestion du temps. Donc le projet d’ISN m’a permis de faire face à cette « contrainte ». Suivant le programme que je devrais réaliser dans le futur, j’ai déjà une idée assez large du temps qu’il pourrait me prendre.

***PS : Avec certains de mes post sur le forum OpenClassRooms j’ai découvert certaines conventions***

 « Pour rappel, on utilise le UpperCamelCase pour nommer les Classe (Ex: CeciEstUneClasse) et le snake\_case pour nommer les méthodes/fonctions/variables (Ex: ceci\_est\_la\_methode\_add). » ***message de Olygrim***

# Diffusion du projet

Notre site perso est à ce jour encore en développement, cependant il est presque terminé. Le projet pacboy et d’autres contenus y seront disponibles.

Ayant fait des recherches sur les licences, en diffusant mon jeu sur Internet j’opterai pour une License « Creative Commons ». Premièrement pour protéger mes droits d’utilisateur, en signant le programme : ainsi le créateur ne peut être usurpé. Ensuite, il est possible que je ne souhaite pas voir une version commercialisée du jeu, c’est pourquoi la License CC est très utile et simple d’utilisation.

# Annexes

**Livre (très utile) :** Apprendre à programmer avec Python 3 de Gérard Swinnen

**Son :**

<http://stackoverflow.com/questions/28127730/how-to-install-pygame-on-python-3-4>

<https://openclassrooms.com/courses/interface-graphique-pygame-pour-python/le-son-5>

<http://www.universal-soundbank.com/>

<http://toutestquantique.fr/laser/>

**Python :**

<http://apprendre-python.com/page-tkinter-interface-graphique-python-tutoriel>

**Autres :**

<https://fr.wikipedia.org/wiki/Loi_des_cosinus>

<https://openclassrooms.com/forum/>

**Logiciels utilisés :**

Paint – Photo filtre – Audacity – Virtual DJ – Gimp – La boîte à couleur

**Programme « Shuriken » (1er prototype)**

[**https://www.dropbox.com/sh/x58lxpcv52ri9d3/AABu9JD4v-DPJhpnzHFtJri8a?dl=0**](https://www.dropbox.com/sh/x58lxpcv52ri9d3/AABu9JD4v-DPJhpnzHFtJri8a?dl=0)