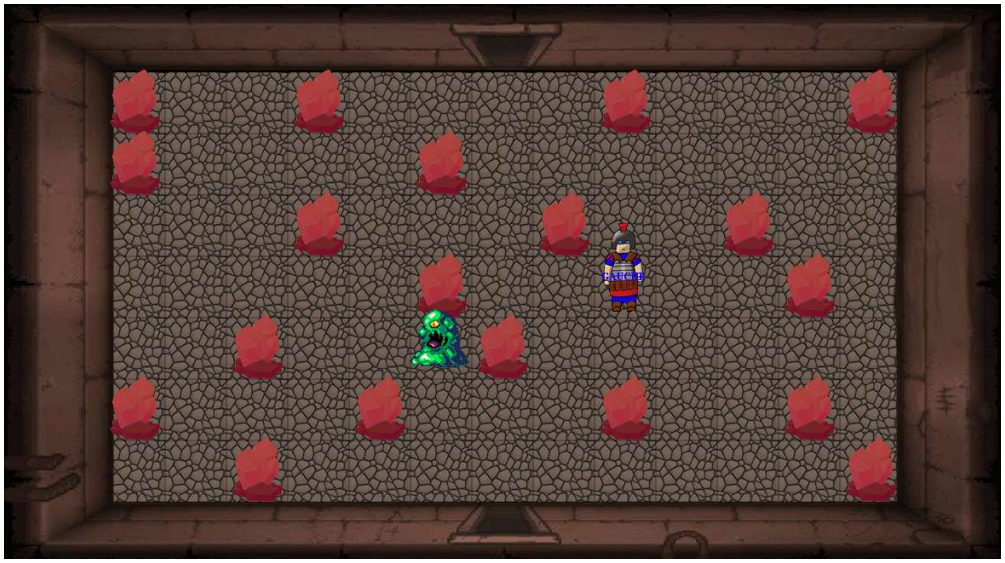
Projet ISN

- SARTORI Tom
- Lycée Albert Schweitzer
- Année 2017/2018
- Professeur d'ISN : Corinne Kesser
- Groupe : Maurer Axel, Seiler Charles, Sartori Tom

Sommaire

Principe du jeu

- Labyrinthe en 13x7
- Graphismes de The Binding of Isaac
- 21 de déplacements par salle
- Monstres possibles



- Idée du projet dans l'année
- Améliorations pour le projet

Notre fonctionnement

- Idées sur papier
- Partage d'idées sur conversation
- Aides en vocal sur discord
- Ajout dans le programme principal
- Programme mit sur dropbox

Avancement du projet

1. Travail en groupe sur papier

2. Déplacement du personnage et début du graphisme

3. Incorporation des menus, de la touche échap et gestion liée au déplacement

4. Ajout du monstre, gestion complète du blocage, améliorations graphiques

Répartition des taches

Charles

- Partie graphique
- Code et gestion des menus

Axel

- Code des éléments de base
- Gestion de certains déplacements du personnage
- Code de l'orientation du personnage

Tom

- Code principal de toutes les autres parties
- Code du blocage et du monstre
- Aide de Charles et Axel

- MenuPrincipal() : affiche les boutons BJouer, BChrono, BHard, BQuitter et modifie AffichagePri

- InitPerso() : lancé à chaque entrée dans une salle

  - initialise *x*, *y*, *pas* et *passage*

  - lance DecomptPas() et InitBlocage()

- haut(event) : lancé avec la fleche du clavier

  - Modifie *x* et *y* pour déplacer le perso avec can.coords()

  - Modifie *passage* pour aller d'une salle à l'autre

  - Change l'image du perso en PersoDos

- InitBlocage() : permet de bloquer des cases pour faire labyrinthe

- IA() : crée un monstre qui se déplace en direction du personnage

  - coordonnées xIA yIA modifiés lorsque x et y changent suivant Rx et Ry

Logiciels utilisés

- Analyse de code

- Format d'images

- Croquis des images

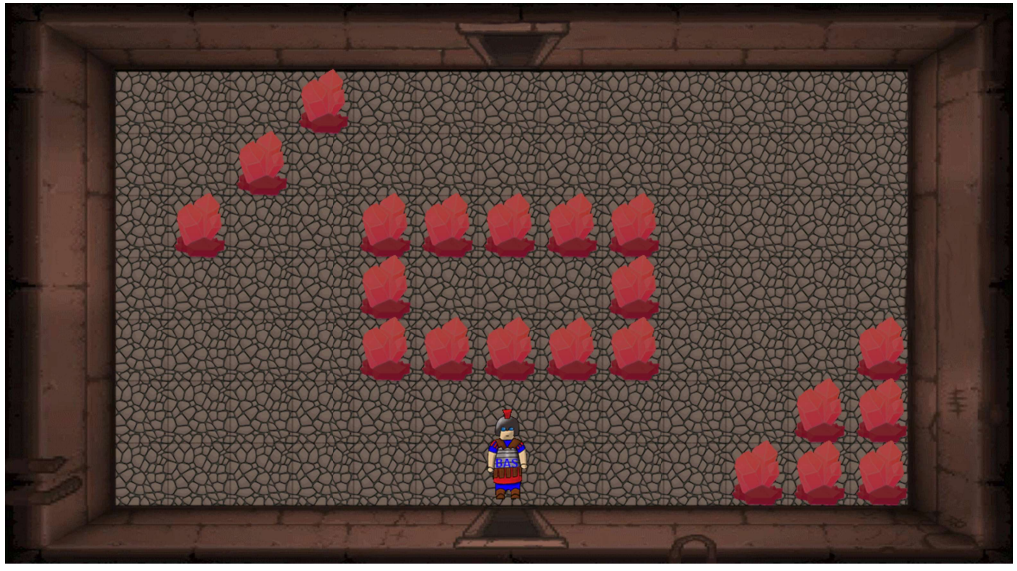
Explication du blocage

- Blocage en trois parties :

  - InitBlocage(), pour afficher les cases bloqués

  - CasePerso(), pour vérifier le blocage ou non de la prochaine case du perso

  - haut(event), pour déplacer ou non le perso



```
def InitBlocage():  # on lance InitBlocage a chaque entrée dans une salle pour initialiser les cases bloqués
    global salle, a, xBloc, ybloc, bloc1, s

    LBloc = [0] * 91  # initialisation des 90 cases sans blocage
    s = []

    s1 = [3, 15, 27, 30, 31, 32, 33, 34, 43, 47, 56, 57, 58, 59, 60, 64, 76, 77, 88, 89, 90]  # index des cases bloqués en salle 1
    s2 = [16, 19, 20, 21, 24, 26, 27, 29, 31, 43, 47, 51, 52, 58, 60, 61, 62, 67, 69, 70]
    #...
    s7 = []

    if salle == 1:
        s = s1
    if salle == 2:
        s = s2
    #...
    if salle == 7:  #salle 7, blocage aléatoire
        for i in range (27):
            x = randint(0, 90)
            if x != 6 and x != 84:
                s7.append(x)
        s = s7

    s.sort()    #permet de mettre la liste dans l'ordre croissant

    xBloc = 0
    yBloc = 0

    for i in range(len(s)): # placement des images du blocage
        a = s[i]    # a change à chaque tour dans la boucle et est égale à la valeur de s à l'indice i
        LBloc[a] = 1

        if a <= 12:     # première ligne
            yBloc = 242

        if a >= 13 and a <= 26:     # deuxième ligne
            yBloc = 314
        #...
        if a >= 79 and a <= 91:     # dernière ligne
            yBloc = 674


        for j in range (13):    #pour les colones
            if (a % 13) == j:
                xBloc = 155 + j*72

        bloc1 = can.create_image(xBloc, yBloc, image=ImBloc1)
```

```python
def haut(event):
    global x, y, v, passage, pas, AffichageIng, AffichagePri, yFut, xFut, case, s, PersoDro, PersoDos, PersoGau, PersoHaut, ImPersoHaut

    xFut = x
    yFut = y - v  # yFut est le coordonnée en y que le perso devrait avoir apres son déplacement
    CasePerso()

    if case in s:  # si la futur case du perso est dans la liste des cases bloqués alors on sort de la fonction pour que le perso n'avance pas
        return ()

    #... suite de la procédure avec la partie du déplacement



def CasePerso():
    global xFut, yFut, case

    case = -1

    for j in range(7):
        for i in range(13):
            if yFut == 228 + j * 72 and xFut == 153 + i * 72:
                case = i + j * 13
```

Difficultés

- Travail de groupe
- Continuité entre les procédures
- Interface graphique
- Blocage du personnage
- Optimisation de certaines parties

Aides

- openclassrooms.com
- developpez.com
- apprendre-python.com
- python.org

Futures améliorations

- Meilleures graphismes
- Ajout d'une monnaie et d'un magasin dans le jeu
- Nouveaux niveaux
- Nouveaux modes
- Classement des meilleures joueurs
- Format du jeu au choix (taille de la fenêtre)

Conclusion

- Projet intéressant
- Nouvelles connaissances
- Meilleure optimisation globale possible
- Futurs ajouts
- Peu de travail d'équipe
- Projet utile pour les prochaines écoles