Elements menu :

- Play : lancer le jeu,
  création d'une room,
  positionement du personnage
- Map : chargement d'un fichier source (.png, .jpeg, image) / photo depuis telephone ou repertoire
  choix de la regle a appliquer / Convertion en noir et blanc selon parametre pixels
  creation d'une liste contenant la map : taille de l'image, nombre de pixels, position de depart, code de la map
  enregistrement d'un fichier .map
  choix d'un fichier .map dans un dossier choix_map

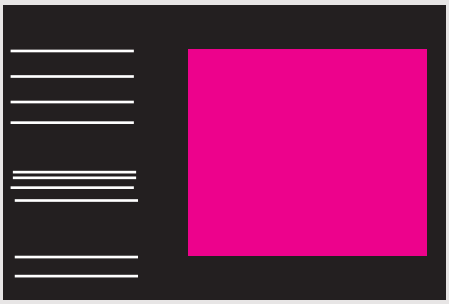
  Option map : regle dynamisme
  ensemble de map

- Setting : Audio : controle du volume (son, music, ect...), activé/désactivé,
  Physique : controle des elements dynamique / gravité, frottement, saut, ect..
  Couleur : choix des couleur
  parametre : full screen / windowed ,
  Commande : changer input

- Credit

- Exit

Map :



choix d'un image depuis le repertoire ou depuis l'appareil photo
visualisation de l'image avec un shader
controle du shader pour déterminer un choix
choix sur la convertion de l'image en binaire
depuis le RGB ou HSB :

```
if ( couleur.r < 150 ) {
    pixels[current] = 1;
} else {
    pixels[current] = 0;
}
```

taille de la map = taille de l'image (ex : 2000x1500)

création de la map depuis l'image
choix d'un nom de map
stockage sur un fichier texte (nom.map)

possibilité d'un choix de regle dynamique
loi :

```
if ( i != ( i - 1 ) ) {
    i = i;
} else {
    i = ( i - 1 );
}
```

Objet et strcuture :

Mouse :
- input
- etat de la souris
- envoie signal
- envoie vecteur

Clavier :
- input
- etat du clavier
- envoie un index

Game :
- variable global
- chargement element
- permanant

Music
- variable musique
- controle musique
- active / deactive
- ensemble des sons

Bouton :
- elements interface
- aspect graphique
- mode d'action
- envoie un signale

Slider :
- elements interface
- aspect graphique
- mode d'action
- envoie une distance

Swap
- elements interface
- aspect graphique
- mode d'action
- envoie un index

Menu :
- creation du menu
- controle du menu
- changement variable
- activé scripte

Menu_pause :
- controle pause/resume
- controle manu
- changement variable

Piece :
- variable physique
- choix de la map
- position du personnage
- création instance

Map :
- variable de la map
- date binaire map
- transformation map
- controle blocs

Personnage :
- ensemble de scripte
- changement scripte
- controle input
- interraction element

Camera :
- deplacment camera
- controle input
- Zoom

Controle_actif :
- zone activation instance
- zone desactivation instance
- suivie personnage

Bloc :
- type de blocs
- affichage
- interraction elements

Menu :

Creation du menu :
- definition de la liste
- creation des element interface

- play
- parametre
- map
- credit
- exit

- audio
- volume / actif
- effet / musique / son
- input
- choix commande
- resolution
- plein ecran
- choix couleur
- affichage actif
- debug affichage
- modification force physique
- debug calibrage variable
- choix PATH du dossier
- divers mode

creation ds_grid
convertie en array
creation des elements
mise en page

controle souris et clavier:
- ciblage par pointeur
- curseur controler par touche
- touche entrer ou click
- switch entre l'un ou l'autre

mise en page :
- scpite de position
- reglage taille
- reglage couleur
- construction animation

- Stockage des parametre dans un fichier texte
- mise a jour des parametre a l'ouverture du menu depuis le fichier
- acces a un dossier contenant les map
- acces a un dossier contenant les photo

nom / type elements / scipte / variables

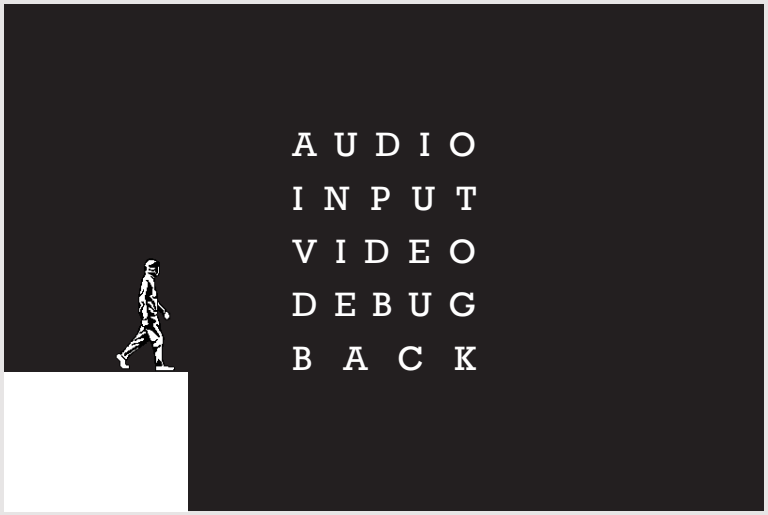
type elements :
- texte
- texte bouton
- bouton lock / unlock / click
- slider lock / unlock
- swap slider / bouton
- affichage image

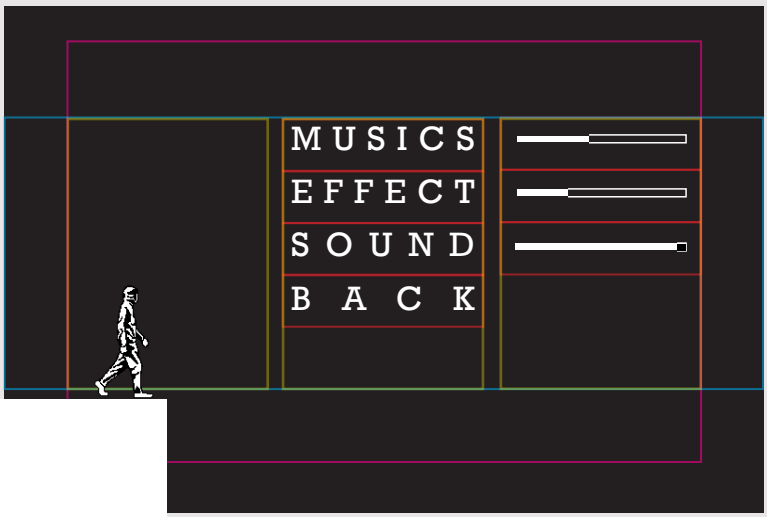
scripte :
- changement page
- modification variable
- load / unload
- play game

variables :
- nombre prédinit selon type elements
- variable local a l'element

- *"bouton" / type.Bouton / change_variable || run_script / 0 / ["on" , "off"] / mode.Lock*
- *"slider" / type.Slider / change_variable / 0.5 / [min, max] / mode.Unlock*
- *"swap" / type.Swap / change_variable / 0 / ["PREMIER", "DEUXIEME", "TROISIEME"] / mode.Ligne*
- *"texte" / type.texte / get_variable*

```
[ page ][ elements ][parametre] = info
[ 0 ][ 0 ][ 2 ] = change_variable( variable );
```



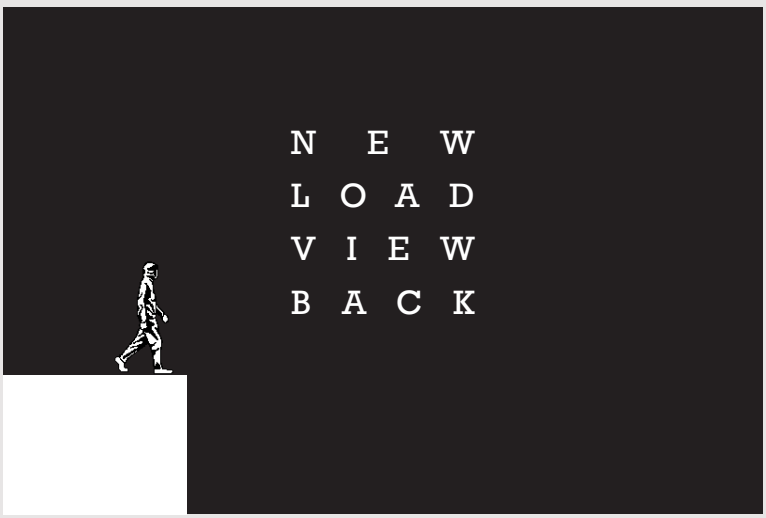


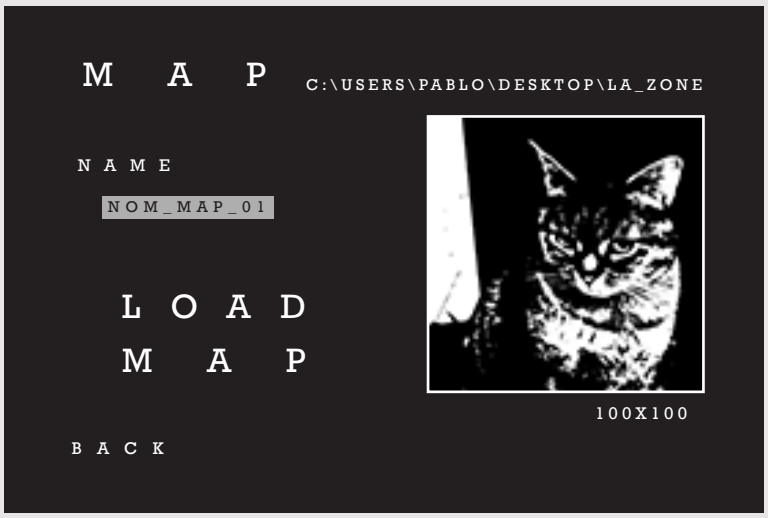
[0] "MUSICS" / type.slider / change_volume_musics / 0.4 / [0, 1] / mode.unlock

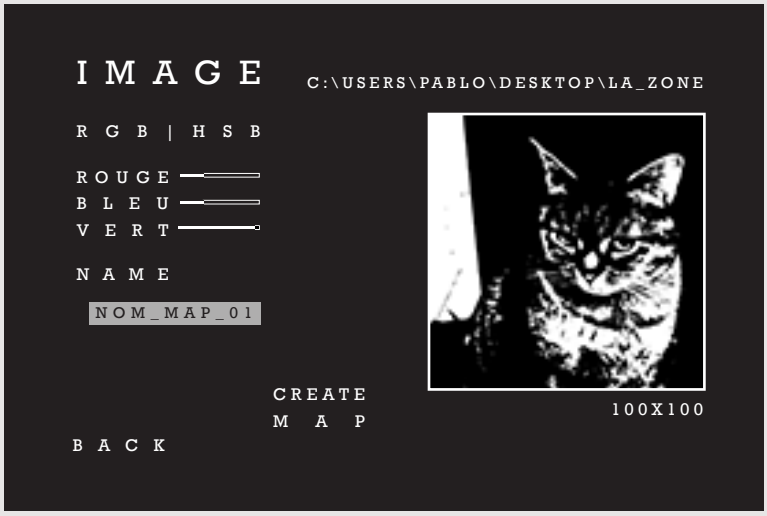
[1] "EFFECT" / type.slider / change_volume_effect / 0.2 / [0, 1] / mode.unlock

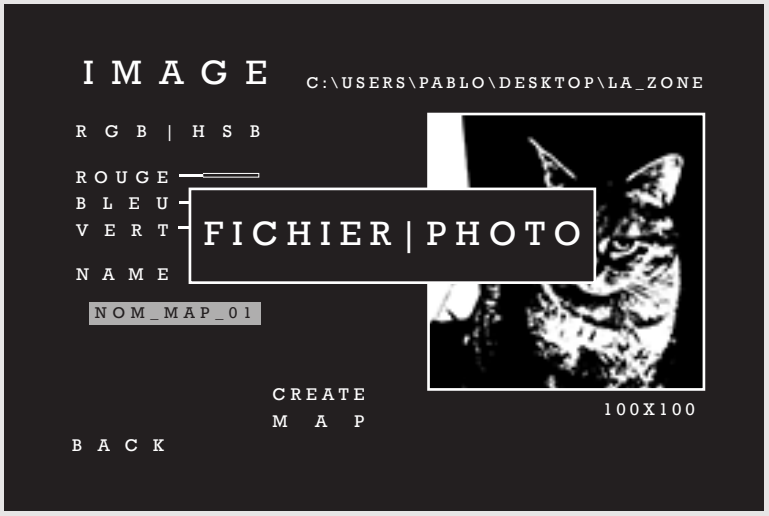
[2] "SOUND" / type.slider / change_volume_sound / 0.9 / [0, 1] / mode.unlock

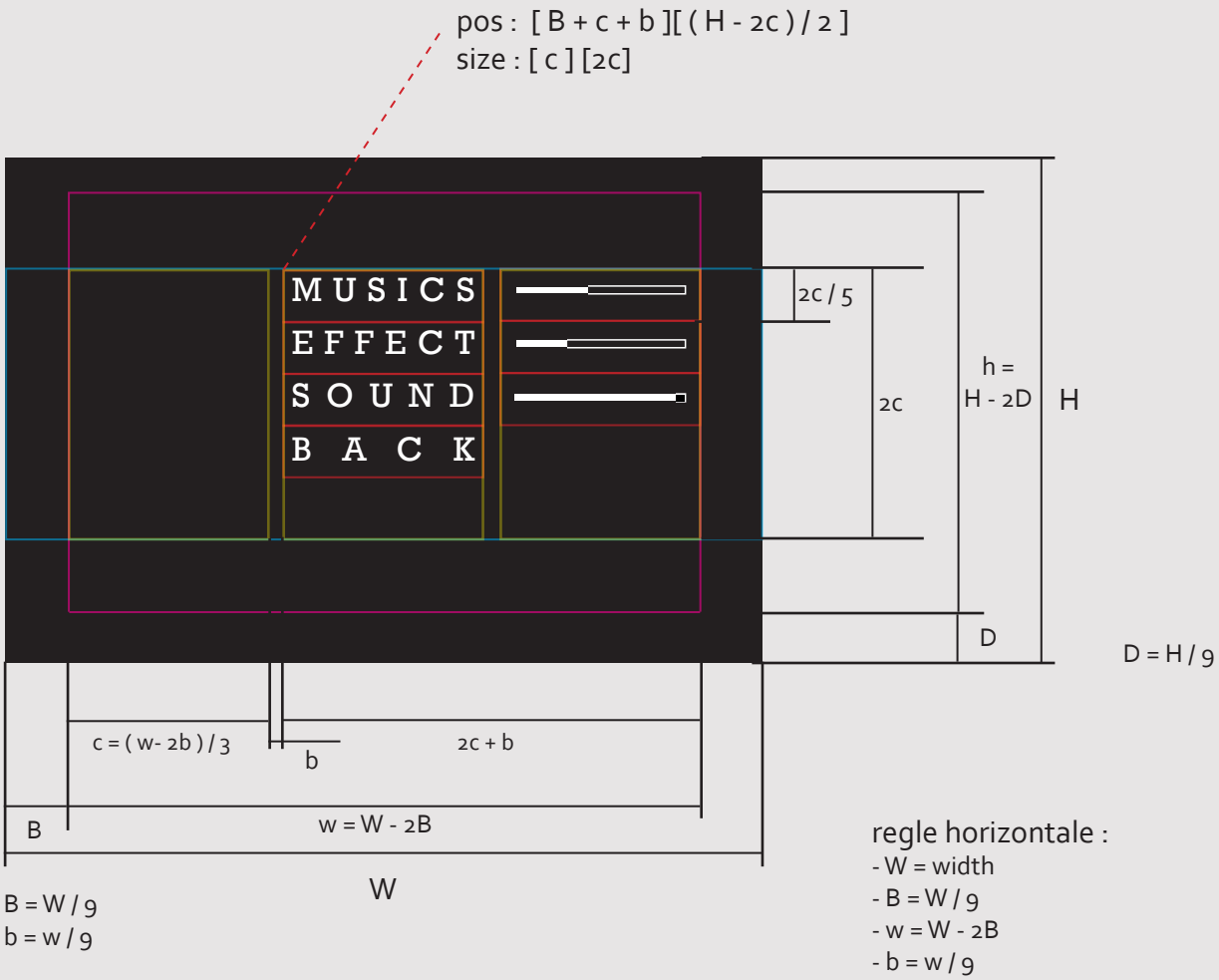
[3] "BACK" / type.bouton / change_page / 0 / [-1] / mode.click











regle horizontale :
- W = width
- B = W / 9
- w = W - 2B
- b = w / 9
- c = (w - 2b) / 3

Structure :
menu_pages[page]
page[option][elements]

regle verticale :
- H = height
- D = H / 9
- h = H - 2D
- e = 2c
- f = 2c / 5

Map :

- nom : " NOM"
- taille : [1200, 1000]
- position depart : [154, 135]
- code map : liste[0, 1, 1, 0, 0, 1, ..., 0, 1]

parametre :
- taille des blocs
- taille de la piece
- position personnage
- regle dynamique

creation blocs :
- position
- type
- etat

Blocs :
bloc blanc / bloc noir
- grimpe disponible
- position
- type
- actif / inactif

tranformation :
- regle definie
- scripte
- nextListe = liste

Repertoire :
- Dossier specifique
- creation fichier
- chargement fichier
- creation si inexistant

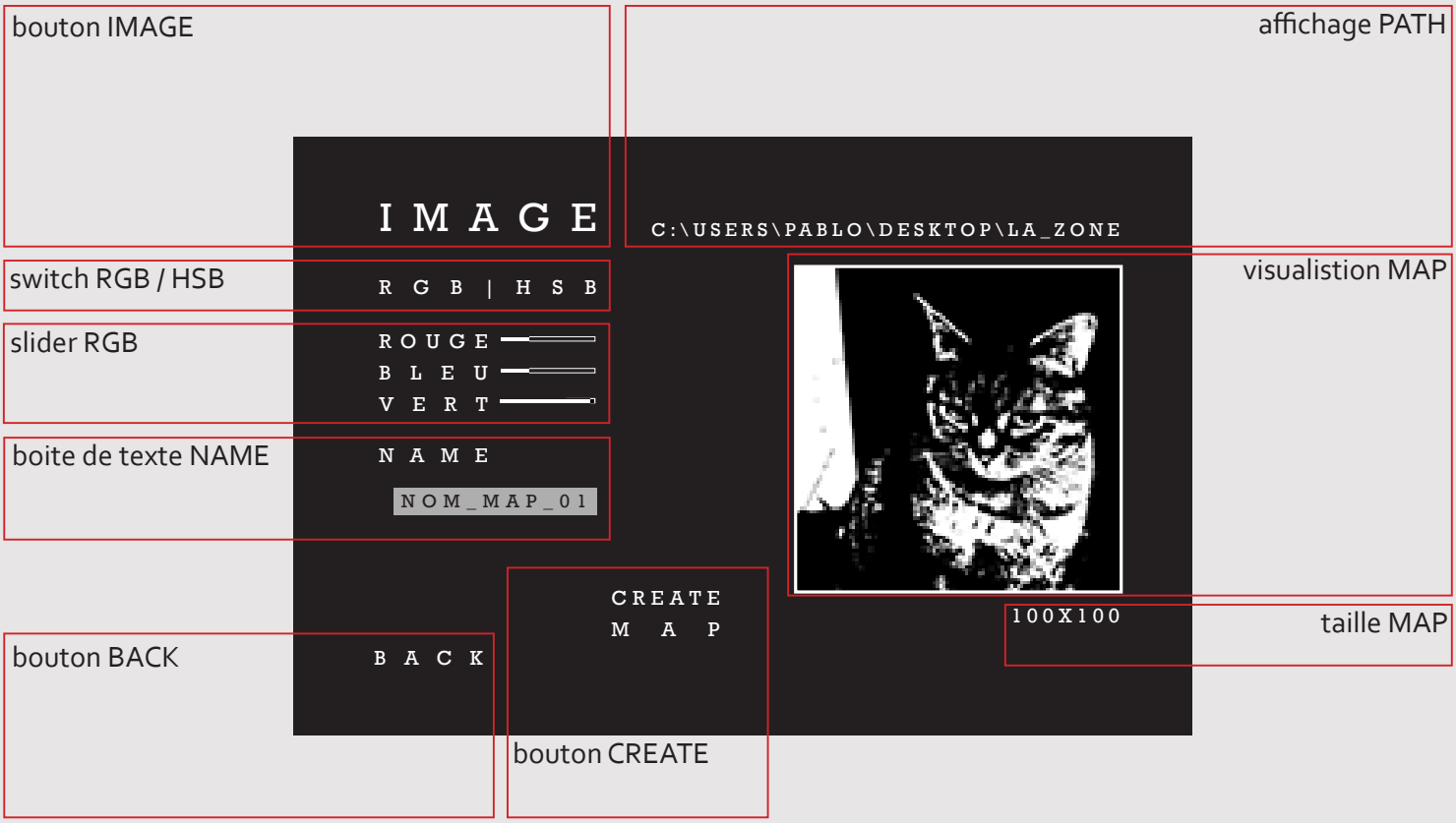
- bouton IMAGE : choisir une photo / definir un fileName ou path
- chargement de l'image
- visualisation de l'image
- shader dynamique : convertie l'image en binaire
- slider RGB : reglage du filtre
- boite de texte NAME : choix d'un nom
- bouton CREATE : crée et enregistre pixel[]
- création d'un fichier name.map

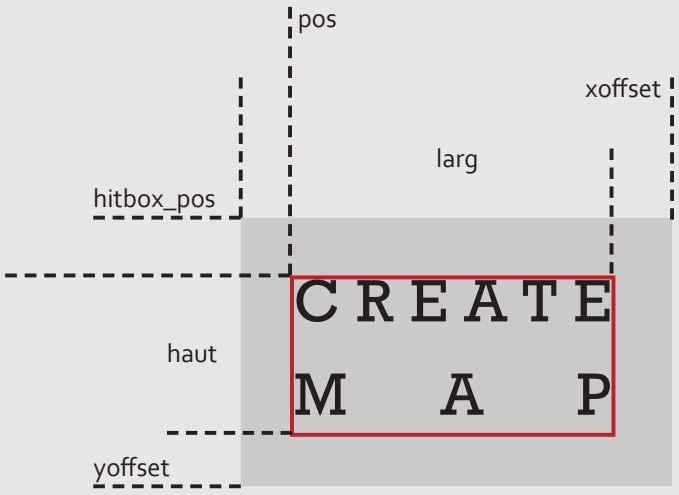
Path : *"C:\Users\Pablo\Pictures\0001.png"*

Img : index vers un sprite

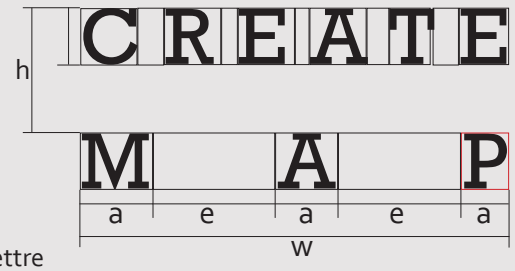
pixel[] : liste binaire, code la map

fonction :
- choisir un path
- charger l'image
- convertir les pixels en binaire





pos[x, y]
size[larg, haut]
texte = ["create", "map"]
texte_font
texte_size
hitbox_pos[x, y]
hitbox_size[larg, haut]
color[stroke, fill, texte]
xoffset
yoffset
mode { lock, unlock, click }
click_on
numero
objet_menu



h = hauteur ligne
w = largeur ligne
a = largeur lettre
nb = nombre de lettre
l = a + a + ... + a
e = (w - l) / (nb - 1)

Bloc

```
if ( pixels[ i ] == o ) {
   create( oBloc_blanc );
} else {
   create( oBloc_noir );
}
```


```
pos = [ X, Y ];
enum type {
  blanc,
  noir
}
largeur = 150;
hauteur = 150;
```

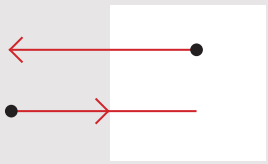
oBloc_blanc

oBloc_noir

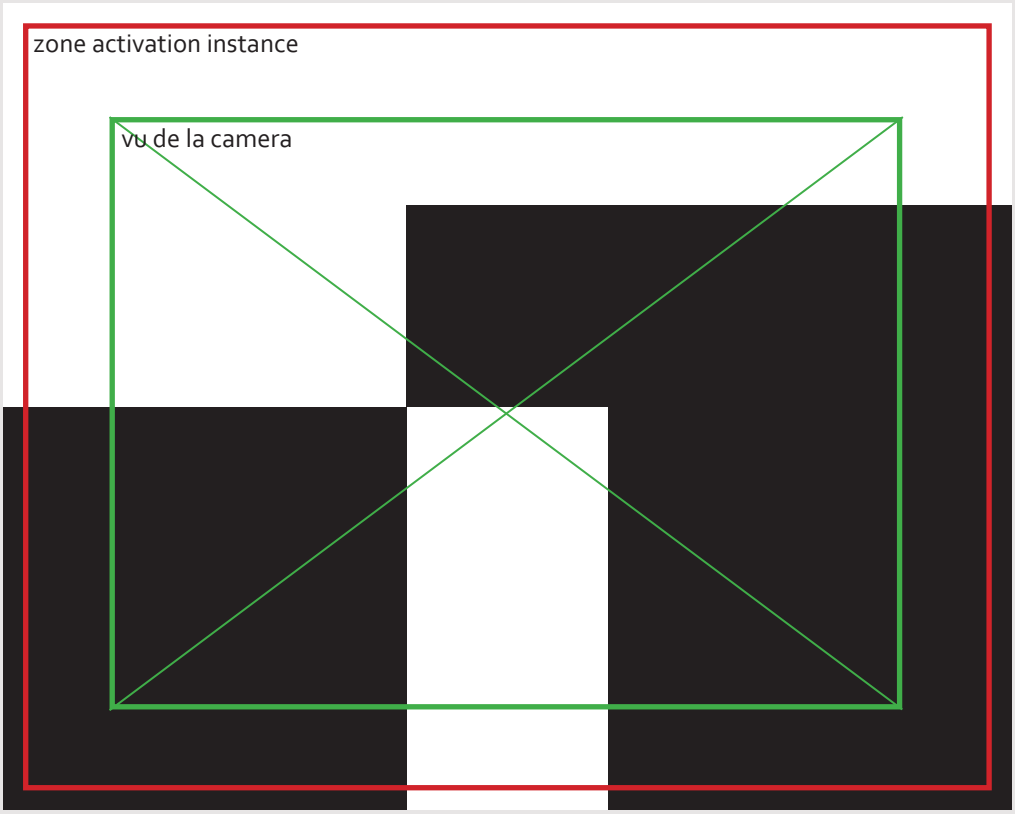
affichage :
- actif si mode noir / blanc
- dessine si proche de la camera
- sprite fix

collision :
- corrige la vitesse la vitesse si l'objet est a l'exterieur du bloc

Interaction :
- collision avec le joueur
- modification parramettre : pos, taille, type
- destrcution / changement couleur
- declanchement animation
- physique au sein du bloc



Camera



- pos = [X, Y]
- resolution = [width, height]
- distance = zoom in/out
- follow = objet

interaction :
- zoom in / out
- suis le joueur
- vitesse de mouvement
- effet controle camera

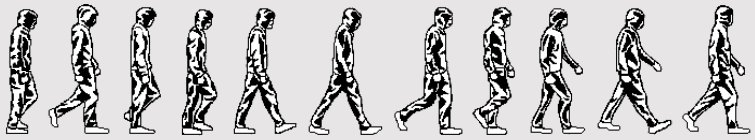
Controle_actif :
- desactive instance en dehors du cadre
- active instance a l'interrieur du cadre
- lance scripte quand camera sort de la zone

Personnage

function :
- play_animation (sprite, start_frame, last_frame, frameRate)
- change_frameRate(variables, frameRate)
- stop / resume_animation()
- goto_frame(sprite, frame)

Spirte :
- frame
- frameCount
- nombre de frame
- frameRate
- fin animation
- pause animation

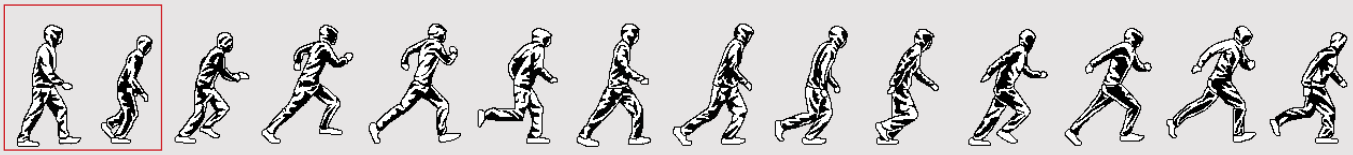
Marche



- 11 frame [0, 10]
- frameRate selon vitesse
- frameRate minimum : 6
- frameRate maximum : 20

- si acceleration.MARCHE
- si sur_le_sol
- si vitesse > 0
- si pas_contre_un_mur

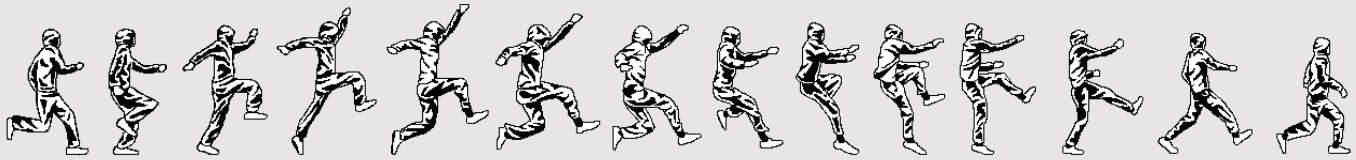
Course



- 14 frame[0, 13]
- frameRate selon vitesse
- frameRate minimum : 7
- frameRate maximum : 20
- frame[0, 1] a la premier boucle selement

- si acceleration.RUN
- si sur_le_sol
- si vitesse > 2
- si pas_contre_un_mur

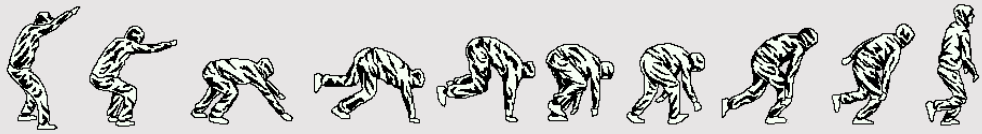
Saut



- 14 frame[0, 13]
- frameRate selon vitesse
- frameRate minimum : 2
- frameRate maximum : 20

- si dans_les_air
- si pas_contre_un_mur

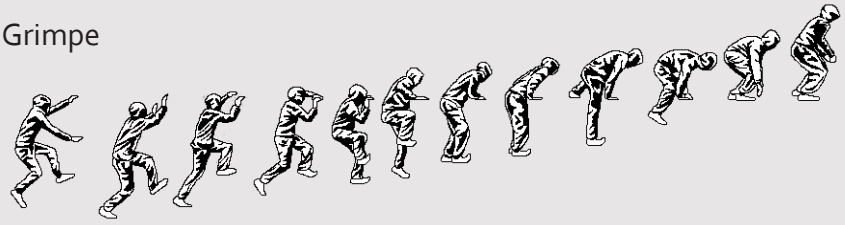
Atterissage



- 10 frame [0, 9]
- frameRate selon vitesse
- frameRate minimum : 6
- frameRate maximum : 13
- fin apres une boucle

- si debut_chute > 200
- si sur_le_sol
- si atterissage.SIMPLE

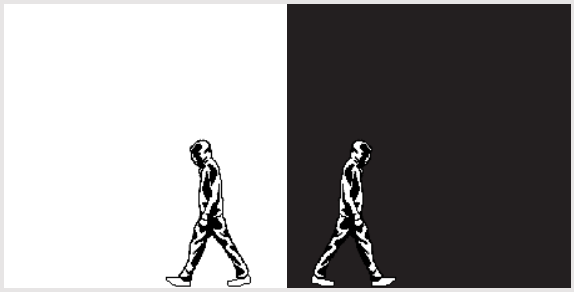
Grimpe



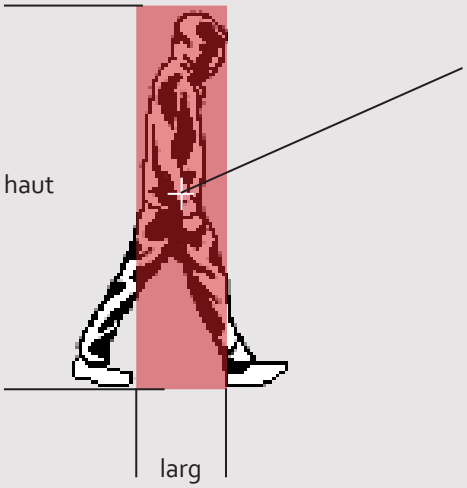
- 12 frame [0, 11]
- frameRate selon vitesse
- frameRate minimum : 6
- frameRate maximum : 12
- fin apres une boucle

- si contre_un_mur
- si accrocher_mur
- si grimpe_on

image_xscale = moveX



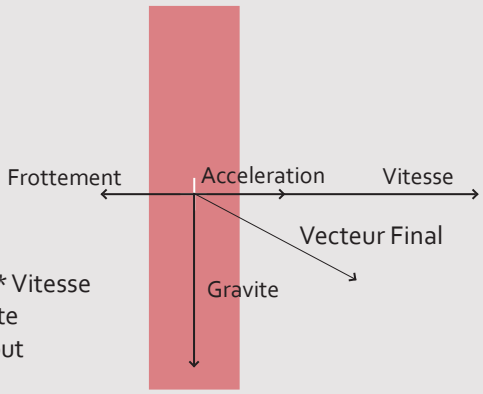
mark_collision



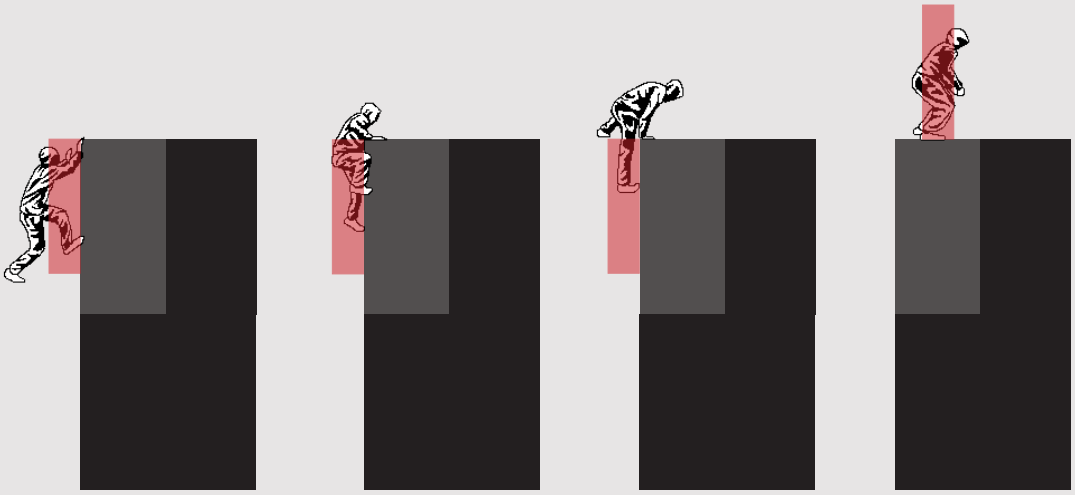
orginie : [X, Y]

X = larg / 2
Y = haut / 2

Vecteur = A + V + G + F



Frottement = 0.x * Vitesse
Gravite = constante
Acceleration = input
Vitesse = cumule



grimpe_dispo
- gauche.on
- droite.off

Fin animation = changement pos

Input :
- Up
- Down
- Left
- Right
- Jump
- Run
- Special

Variable :

pos[X, Y]
vitesse
force
gravite
frottement
acceleration[marche, course]
force_jump

Physique :

frottement sol / air
densité air
force gravite
force
resistance

Fonction :

collision_bloc()
add_vitesse(vitesse)
choix_sprite()

Etat :

Acceleration { marche, run };
dans_les_air
jump_dispo
sur_le_sol
contre_un_mur
grimpe_on
Atterissage{ simple, roulade }
jump_on
debut_chut

lecture input :
- moveX = right - left
- moveY = down - up
- jump_on = key_jump
- run_on = key_run
- pause_on = key_pause

calcul de la vitesse :
- vit = V + F + A + G

correction de la vitesse :
- collision

deplacement:
- ajout de la vitesse

choix d'un etat :
- determine un choix

choix d'un sprite :
- recuperation donnée

calcul animation :
- changement de frame