RIVIERE Geoffrey

CAPART Mathis

SOBIERAJ Elio

Projet de fin d'année

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

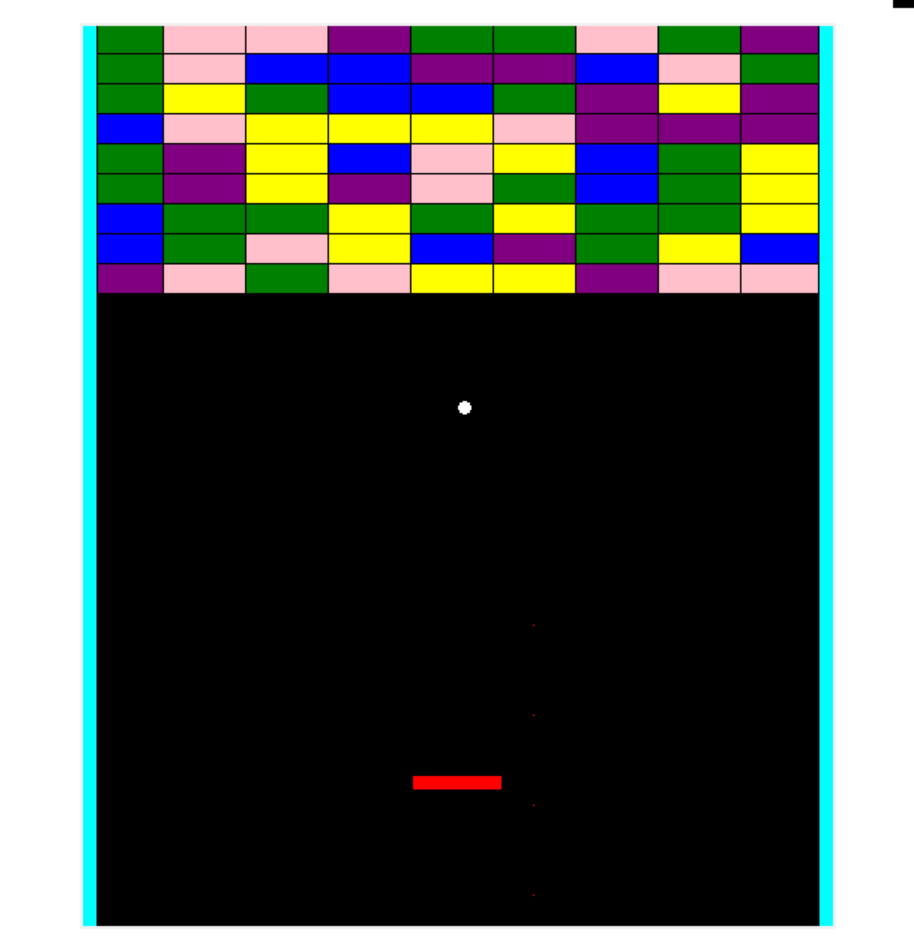
Documentation technique :

LE JEUX :

Bienvenu pour la présentation de la documentation technique de la derniere sortie de Game Unleashed, depuis 1978 nous nous sommes spécialisé dans la créations de jeux retro qui plaisent aux petits comme aux grands.

Sur cette présentation passerons étapes par étapes chaque fonctionnalitées de break-it, le dernier exploit de la companie.





LE TABLEAU :



Le tableau du jeux est par des dimention de 9 brique de haut pour 9 briques de large,

Les briques sont quant à elle crée selon un nombre de vie défini aléatoirement selon un code couleur

Bleu = 5 vies

Vert = 4 vies

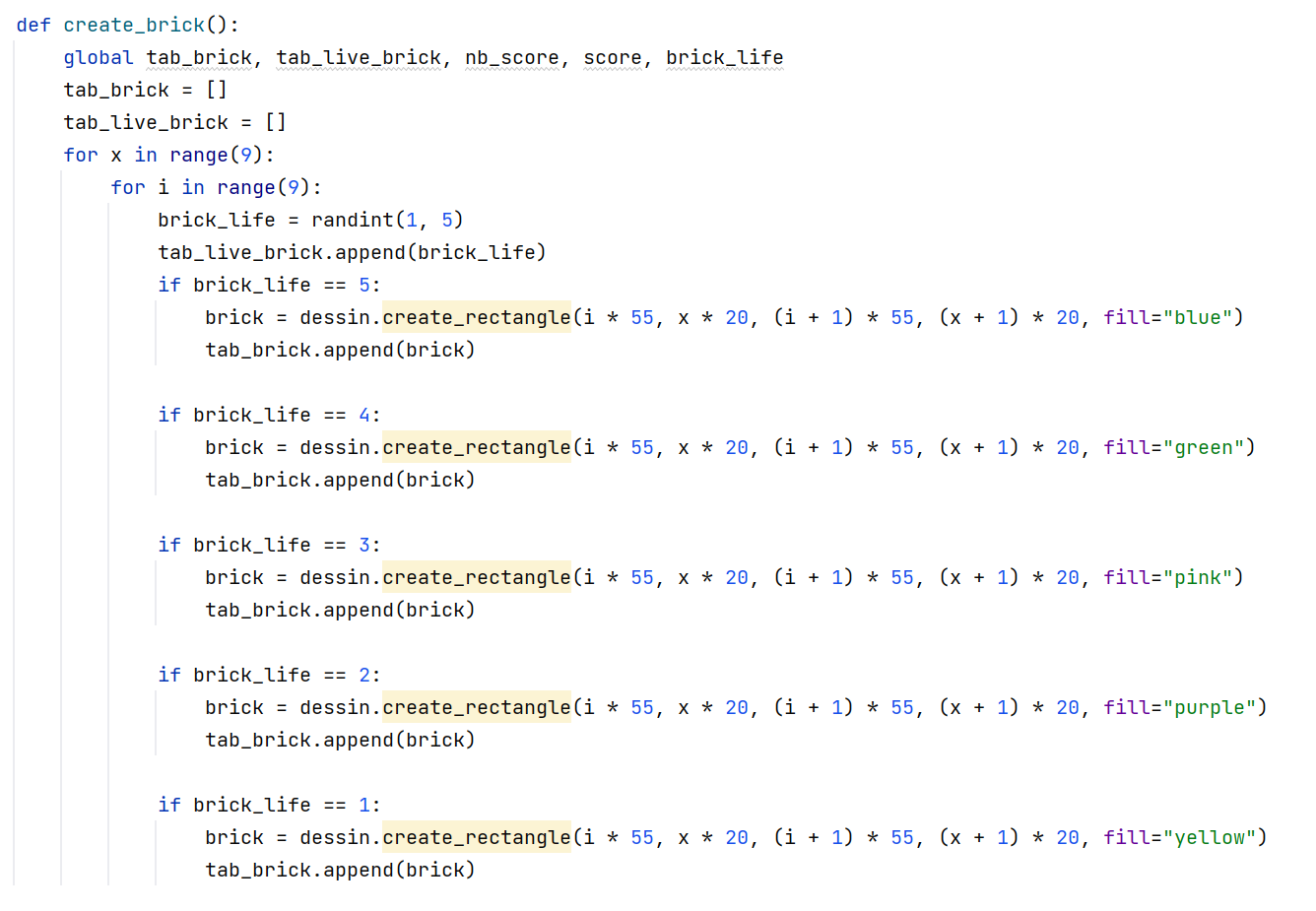
Rose = 3 vies

Violet = 2 vies

Jaunes = 1 vie

lorsque la brique n'a plus de vie alors cette dernière est détruite.

Le tableau est défini par la fonction create\_tab\_brick ci-dessous

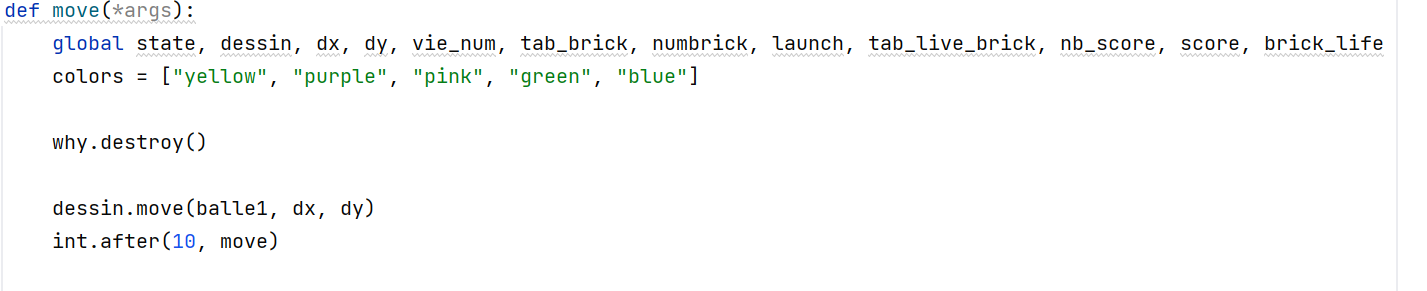


LE MOUVEMENT :

Le mouvement dans le jeux est présent que dans 2 élément , la plateforme et la balle

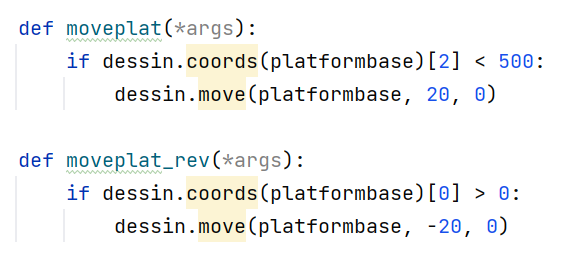
Le mouvement de la balle et définie par deux variable "dx" sa vitesse horizontale, et "dy" sa vitesse verticale.

Le mouvement de la balle est défini par une partie de la fonction move ci dessous



La plateforme et quant a elle déplaçable sur l'axe horizontale grace au flèche directionnel droite et gauche et ne peux dépasser les murs a droite et a gauche du tableau.

Le mouvement de la plateforme est défini par les fonction moveplat qui deplace a droite la plateforme et moveplat\_rev qui déplace la plateforme a gauche



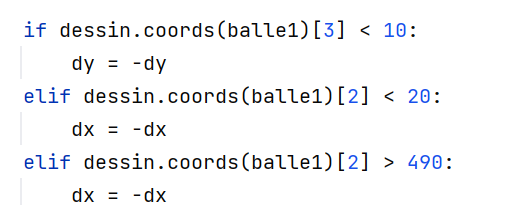
le changement directionnel de la balle est provoqué par plusieur facteurs :

- si il a contacte avec les murs ou le plafond

- si il a contacte avec la plateforme

Si il y a un contacte entre la balle le plafond alors la coordonée verticale de la balle est inversé , si il y a contacte entre la balle et les murs alors c'est la coordonée horizontale qui est invérsé

ceci est défini par la partie ci dessous de la fonction move



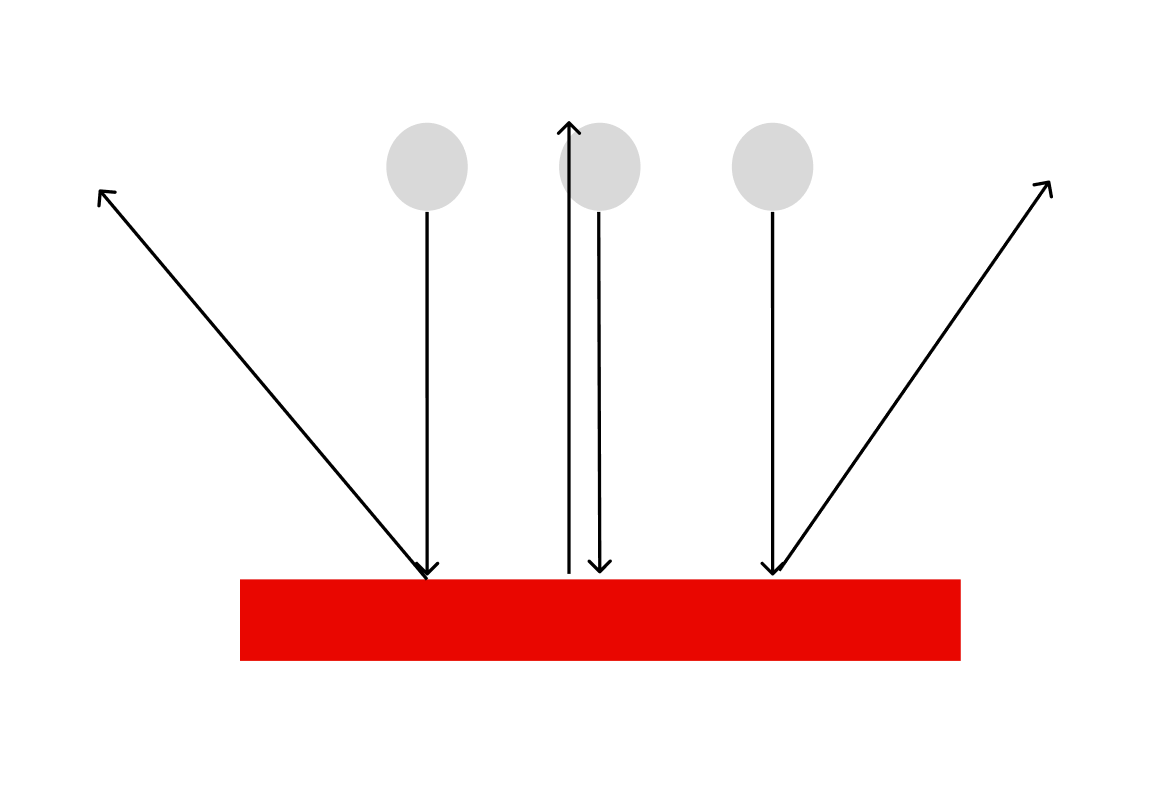
La collisions de la balle avec la plateforme est differente, celle si suivant certaines règles,

- si la balle touche la partie centrale de la plateforme alors la balle est retourné droite

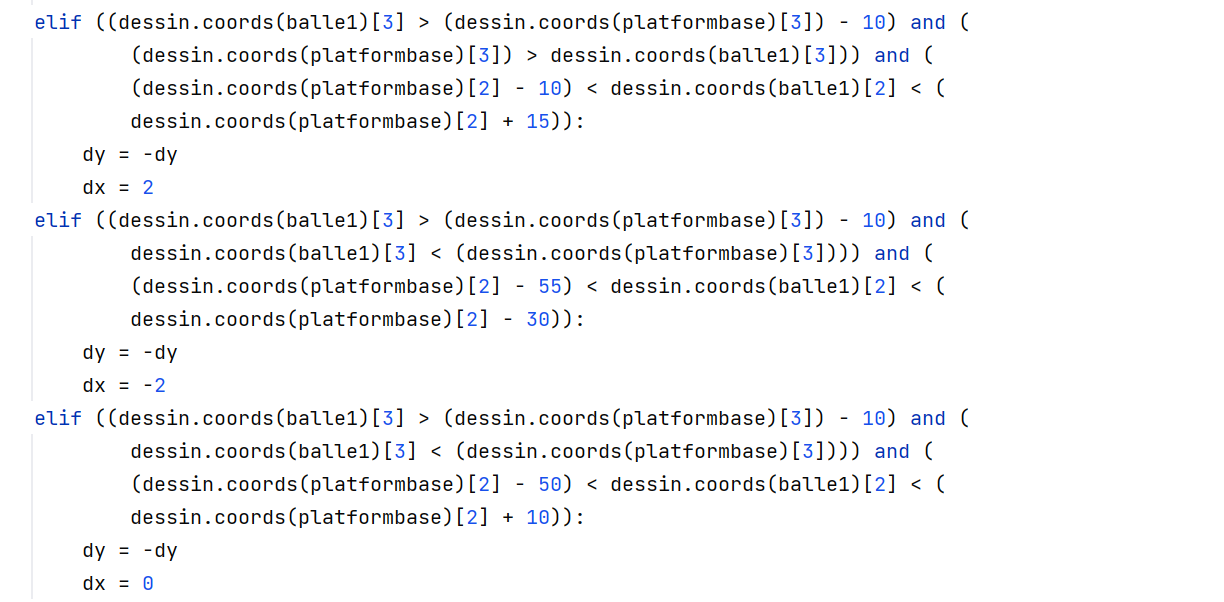
- si la balle touche la partie droite de la plateforme alors la balle est renvoyé en diagonale à droite

- si la balle touche la partie gauche de la plateforme alors la balle est renvoyé en diagonale à gauche

ces règles sont éxpliqué par le schéma ci dessous



les collisions de la balle sont défini par la partie ci-dessous de la fonction move



LES COLLISIONS :

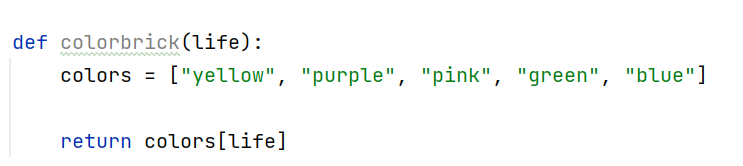
Les collisions entre la balle et les brique peuvent soit :

- entrener une perte de vie pour la brique

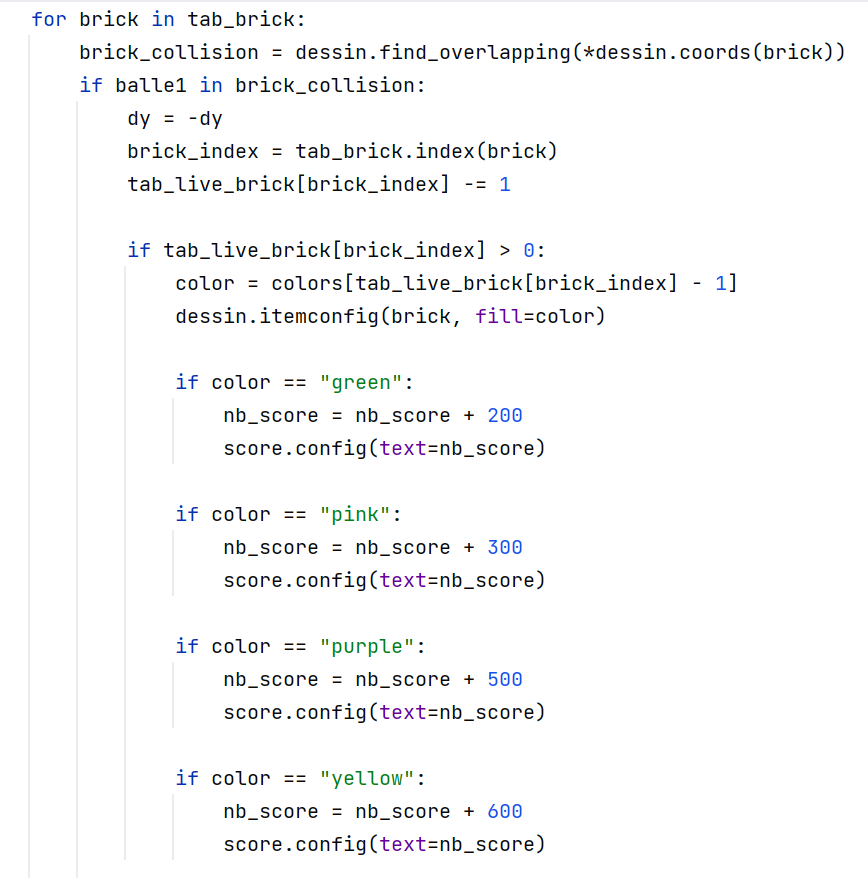
- entrener la destuction de la brique si elle n'as plus de vie

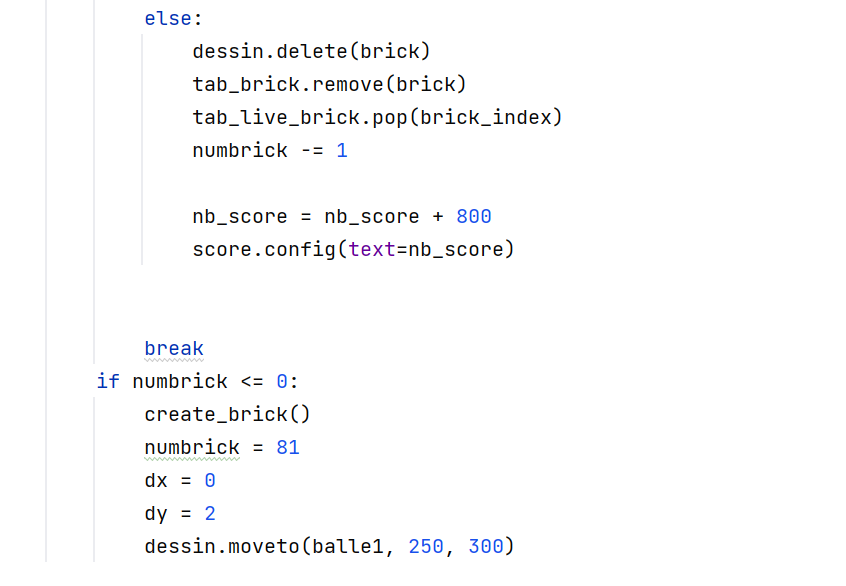
la collision entre la balle et la brique enchaine un changement de la coordonée verticale pour la briques

la perte de vie changeant la couleure de la brique la fonction colorbrique est alors appelé pour définir la couleur attribué au nombre de vie de la brique



les collisions entre les briques et la balle sont géré grace à la partie ci-dessous de la fonction move



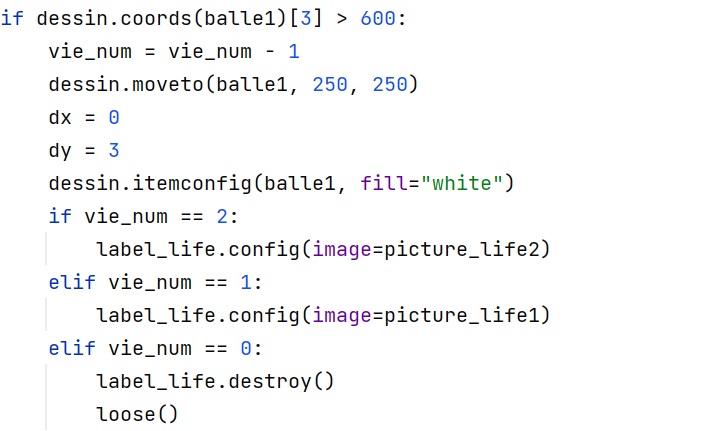


LES EVENEMENTS :

Le premier évènement apparait si les 3 vies sont utilisé c'est alors l'évenement de défaite

lors de l'évenement de défaite on a donc l'affichage du score et un bouton exit pour quitter le tableau de jeu

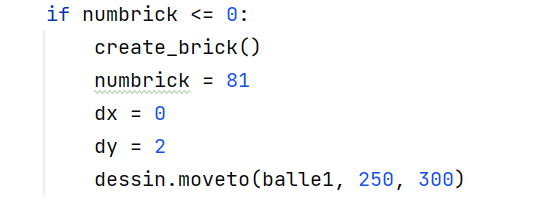
une perte de vie est engendré si la balle arrive en dessous de la bordure basse du tableau de jeu ceci étant géré par la partie ci-dessous de la fontction move



l'évenement de défaite etant quant à lui géré par la fonction loose



le second événement intervient si toutes les brique du tableau sont détruite alors un nouveau tableau aléatoire de brique est crée, ceci étant géré par la partie ci-dessous de la fonction move avec numbrique qui définie le nombre de briques



LE MENU :

Le menu se présente de cette façon :



Ce bouton sert à lancer le jeu :

Une image contenant Police, logo, symbole, Bleu électrique

Description générée automatiquement

start\_game = Button(menu, image=start, command=on\_game, relief="flat", borderwidth=0, bg="white")  
start\_game.pack(pady=10)

Ce bouton sert à afficher le tableau des score :

Une image contenant texte, Police, vert, logo

Description générée automatiquement

btn\_top\_score = Button(menu, text="tableau des score", relief="flat", font=("yellowstone", 20), bg="green", fg="white", command=open\_score)  
btn\_top\_score.pack(pady=10)

Et ce bouton sert a femer le programme :

Une image contenant texte, Police, logo, symbole

Description générée automatiquement

btn\_exit = Button(menu, image=exit, command=exit\_windows, relief="flat", borderwidth=0, bg="white")  
btn\_exit.pack(ipadx=20, pady=5)

LE SCORE :

Pour l’affichage du score, nous avons initialisé le score à 0 au début du script

nb\_score = 0

ensuite, nous avons ajouté un nombre en fonction de la brique qui est touché,

brique bleu touché :

if color == "green":  
 nb\_score = nb\_score + 200  
 score.config(text=nb\_score)

brique vert touché :

if color == "pink":  
 nb\_score = nb\_score + 300  
 score.config(text=nb\_score)

brique rose touché :

if color == "purple":  
 nb\_score = nb\_score + 500  
 score.config(text=nb\_score)

brique violet touché :

if color == "yellow":  
 nb\_score = nb\_score + 600  
 score.config(text=nb\_score)

brique jaune touché :

else:  
 nb\_score = nb\_score + 800  
 score.config(text=nb\_score)

pour affcihé le code nous avons utiliser un label,

score = Label(int, background="black", borderwidth=4, width=5, text=nb\_score, fg="white",font=("yellowstone", 20),)  
score.pack(side="top")

Puis pour l’affichage du score a la fin du jeu nous avons utilisé ceci :

appel\_score = IntVar()  
appel\_score.set(nb\_score)  
  
display\_score = Label(frame\_score, text="Ton Score :", bg="white", fg="black", font=("yellowstone", 20))  
display\_score.pack(side="left")  
  
display\_score\_number = Label(frame\_score, textvariable=appel\_score, bg="white", fg="black",font=("yellowstone", 20))  
display\_score\_number.pack(side="right")

LES VIES :

Pour le système de vie, nous avons d’abord téléchargé 3 images :

picture\_life1 = PhotoImage(file="image/life\_1.png")  
picture\_life2 = PhotoImage(file="image/life\_2.png")  
picture\_life3 = PhotoImage(file="image/life\_3.png")







Ensuite nous avons créer un label pour y coller la première image :

label\_life = Label(int, image=picture\_life3, bg="white")  
label\_life.pack()

puis en fonction des vies on modifie l’image collé au label :

if vie\_num == 2:  
 label\_life.config(image=picture\_life2)  
elif vie\_num == 1:  
 label\_life.config(image=picture\_life1)

Et si les vies sont égales à 0, alors on supprime le label,

elif vie\_num == 0:  
 label\_life.destroy()  
 enter\_name()

L'INFRASTRUCTURE :

Pour l’infrastructure, nous avons opté pour un serveur linux sous Rocky linux pour le web serveur. Puis on a installé httpd, ssh, PHP, mysql-server. Puis nous avons créé un dossier sub\_domain dans var/www pour si plus tard nous utilisons plusieurs sites pour ce serveur. Puis on configure le fichier httpd.conf :

#

# This is the main Apache HTTP server configuration file. It contains the

# configuration directives that give the server its instructions.

# See <URL:http://httpd.apache.org/docs/2.4/> for detailed information.

# In particular, see

# <URL:http://httpd.apache.org/docs/2.4/mod/directives.html>

# for a discussion of each configuration directive.

#

# See the httpd.conf(5) man page for more information on this configuration,

# and httpd.service(8) on using and configuring the httpd service.

#

# Do NOT simply read the instructions in here without understanding

# what they do. They're here only as hints or reminders. If you are unsure

# consult the online docs. You have been warned.

#

# Configuration and logfile names: If the filenames you specify for many

# of the server's control files begin with "/" (or "drive:/" for Win32), the

# server will use that explicit path. If the filenames do \*not\* begin

# with "/", the value of ServerRoot is prepended -- so 'log/access\_log'

# with ServerRoot set to '/www' will be interpreted by the

# server as '/www/log/access\_log', where as '/log/access\_log' will be

# interpreted as '/log/access\_log'.

#

# ServerRoot: The top of the directory tree under which the server's

# configuration, error, and log files are kept.

#

# Do not add a slash at the end of the directory path. If you point

# ServerRoot at a non-local disk, be sure to specify a local disk on the

# Mutex directive, if file-based mutexes are used. If you wish to share the

# same ServerRoot for multiple httpd daemons, you will need to change at

# least PidFile.

#

ServerRoot "/etc/httpd"

#

# Listen: Allows you to bind Apache to specific IP addresses and/or

# ports, instead of the default. See also the <VirtualHost>

# directive.

#

# Change this to Listen on a specific IP address, but note that if

# httpd.service is enabled to run at boot time, the address may not be

# available when the service starts. See the httpd.service(8) man

# page for more information.

#

#Listen 12.34.56.78:80

Listen 80

#

# Dynamic Shared Object (DSO) Support

#

# To be able to use the functionality of a module which was built as a DSO you

# have to place corresponding `LoadModule' lines at this location so the

# directives contained in it are actually available \_before\_ they are used.

# Statically compiled modules (those listed by `httpd -l') do not need

# to be loaded here.

#

# Example:

# LoadModule foo\_module modules/mod\_foo.so

#

Include conf.modules.d/\*.conf

#

# If you wish httpd to run as a different user or group, you must run

# httpd as root initially and it will switch.

#

# User/Group: The name (or #number) of the user/group to run httpd as.

# It is usually good practice to create a dedicated user and group for

# running httpd, as with most system services.

#

User apache

Group apache

# 'Main' server configuration

#

# The directives in this section set up the values used by the 'main'

# server, which responds to any requests that aren't handled by a

# <VirtualHost> definition. These values also provide defaults for

# any <VirtualHost> containers you may define later in the file.

#

# All of these directives may appear inside <VirtualHost> containers,

# in which case these default settings will be overridden for the

# virtual host being defined.

#

#

# ServerAdmin: Your address, where problems with the server should be

# e-mailed. This address appears on some server-generated pages, such

# as error documents. e.g. admin@your-domain.com

#

ServerAdmin root@localhost

#

# ServerName gives the name and port that the server uses to identify itself.

# This can often be determined automatically, but we recommend you specify

# it explicitly to prevent problems during startup.

#

# If your host doesn't have a registered DNS name, enter its IP address here.

#

#ServerName www.example.com:80

ServerName www.break-it.lan:80

#

# Deny access to the entirety of your server's filesystem. You must

# explicitly permit access to web content directories in other

# <Directory> blocks below.

#

<Directory />

AllowOverride none

Require all denied

</Directory>

#

# Note that from this point forward you must specifically allow

# particular features to be enabled - so if something's not working as

# you might expect, make sure that you have specifically enabled it

# below.

#

#

# DocumentRoot: The directory out of which you will serve your

# documents. By default, all requests are taken from this directory, but

# symbolic links and aliases may be used to point to other locations.

#

DocumentRoot "/var/www/html/www.break-it.lan"

#

# Relax access to content within /var/www.

#

<Directory "/var/www">

AllowOverride None

# Allow open access:

Require all granted

</Directory>

# Further relax access to the default document root:

<Directory "/var/www/html">

#

# Possible values for the Options directive are "None", "All",

# or any combination of:

# Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews

#

# Note that "MultiViews" must be named \*explicitly\* --- "Options All"

# doesn't give it to you.

#

# The Options directive is both complicated and important. Please see

# http://httpd.apache.org/docs/2.4/mod/core.html#options

# for more information.

#

Options Indexes FollowSymLinks

#

# AllowOverride controls what directives may be placed in .htaccess files.

# It can be "All", "None", or any combination of the keywords:

# Options FileInfo AuthConfig Limit

#

AllowOverride None

#

# Controls who can get stuff from this server.

#

Require all granted

</Directory>

#

# DirectoryIndex: sets the file that Apache will serve if a directory

# is requested.

#

<IfModule dir\_module>

DirectoryIndex index.html

</IfModule>

#

# The following lines prevent .htaccess and .htpasswd files from being

# viewed by Web clients.

#

<Files ".ht\*">

Require all denied

</Files>

#

# ErrorLog: The location of the error log file.

# If you do not specify an ErrorLog directive within a <VirtualHost>

# container, error messages relating to that virtual host will be

# logged here. If you \*do\* define an error logfile for a <VirtualHost>

# container, that host's errors will be logged there and not here.

#

ErrorLog "logs/error\_log"

#

# LogLevel: Control the number of messages logged to the error\_log.

# Possible values include: debug, info, notice, warn, error, crit,

# alert, emerg.

#

LogLevel warn

<IfModule log\_config\_module>

#

# The following directives define some format nicknames for use with

# a CustomLog directive (see below).

#

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined

LogFormat "%h %l %u %t \"%r\" %>s %b" common

<IfModule logio\_module>

# You need to enable mod\_logio.c to use %I and %O

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %I %O" combinedio

</IfModule>

#

# The location and format of the access logfile (Common Logfile Format).

# If you do not define any access logfiles within a <VirtualHost>

# container, they will be logged here. Contrariwise, if you \*do\*

# define per-<VirtualHost> access logfiles, transactions will be

# logged therein and \*not\* in this file.

#

#CustomLog "logs/access\_log" common

#

# If you prefer a logfile with access, agent, and referer information

# (Combined Logfile Format) you can use the following directive.

#

CustomLog "logs/access\_log" combined

</IfModule>

<IfModule alias\_module>

#

# Redirect: Allows you to tell clients about documents that used to

# exist in your server's namespace, but do not anymore. The client

# will make a new request for the document at its new location.

# Example:

# Redirect permanent /foo http://www.example.com/bar

#

# Alias: Maps web paths into filesystem paths and is used to

# access content that does not live under the DocumentRoot.

# Example:

# Alias /webpath /full/filesystem/path

#

# If you include a trailing / on /webpath then the server will

# require it to be present in the URL. You will also likely

# need to provide a <Directory> section to allow access to

# the filesystem path.

#

# ScriptAlias: This controls which directories contain server scripts.

# ScriptAliases are essentially the same as Aliases, except that

# documents in the target directory are treated as applications and

# run by the server when requested rather than as documents sent to the

# client. The same rules about trailing "/" apply to ScriptAlias

# directives as to Alias.

#

ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"

</IfModule>

#

# "/var/www/cgi-bin" should be changed to whatever your ScriptAliased

# CGI directory exists, if you have that configured.

#

<Directory "/var/www/cgi-bin">

AllowOverride None

Options None

Require all granted

</Directory>

<IfModule mime\_module>

#

# TypesConfig points to the file containing the list of mappings from

# filename extension to MIME-type.

#

TypesConfig /etc/mime.types

#

# AddType allows you to add to or override the MIME configuration

# file specified in TypesConfig for specific file types.

#

#AddType application/x-gzip .tgz

#

# AddEncoding allows you to have certain browsers uncompress

# information on the fly. Note: Not all browsers support this.

#

#AddEncoding x-compress .Z

#AddEncoding x-gzip .gz .tgz

#

# If the AddEncoding directives above are commented-out, then you

# probably should define those extensions to indicate media types:

#

AddType application/x-compress .Z

AddType application/x-gzip .gz .tgz

#

# AddHandler allows you to map certain file extensions to "handlers":

# actions unrelated to filetype. These can be either built into the server

# or added with the Action directive (see below)

#

# To use CGI scripts outside of ScriptAliased directories:

# (You will also need to add "ExecCGI" to the "Options" directive.)

#

#AddHandler cgi-script .cgi

# For type maps (negotiated resources):

#AddHandler type-map var

#

# Filters allow you to process content before it is sent to the client.

#

# To parse .shtml files for server-side includes (SSI):

# (You will also need to add "Includes" to the "Options" directive.)

#

AddType text/html .shtml

AddOutputFilter INCLUDES .shtml

</IfModule>

#

# Specify a default charset for all content served; this enables

# interpretation of all content as UTF-8 by default. To use the

# default browser choice (ISO-8859-1), or to allow the META tags

# in HTML content to override this choice, comment out this

# directive:

#

AddDefaultCharset UTF-8

<IfModule mime\_magic\_module>

#

# The mod\_mime\_magic module allows the server to use various hints from the

# contents of the file itself to determine its type. The MIMEMagicFile

# directive tells the module where the hint definitions are located.

#

MIMEMagicFile conf/magic

</IfModule>

#

# Customizable error responses come in three flavors:

# 1) plain text 2) local redirects 3) external redirects

#

# Some examples:

#ErrorDocument 500 "The server made a boo boo."

#ErrorDocument 404 /missing.html

#ErrorDocument 404 "/cgi-bin/missing\_handler.pl"

#ErrorDocument 402 http://www.example.com/subscription\_info.html

#

#

# EnableMMAP and EnableSendfile: On systems that support it,

# memory-mapping or the sendfile syscall may be used to deliver

# files. This usually improves server performance, but must

# be turned off when serving from networked-mounted

# filesystems or if support for these functions is otherwise

# broken on your system.

# Defaults if commented: EnableMMAP On, EnableSendfile Off

#

#EnableMMAP off

EnableSendfile on

<VirtualHost \*:80>

ServerName www.break-it.lan

ServerAdmin username@rockylinux.org

DocumentRoot /var/www/html/www.break-it.lan/

DirectoryIndex index.php index.htm index.html

Alias /icons/ /var/www/icons/

# ScriptAlias /cgi-bin/ /var/www/sub-domains/com.ourownwiki.www/cgi-bin/

CustomLog "/var/log/httpd/www.break-it.lan-access\_log" combined

ErrorLog "/var/log/httpd/www.break-it.lan-error\_log"

<Directory /var/www/html/www.break-it.lan/>

Options -ExecCGI -Indexes

AllowOverride None

Order deny,allow

Deny from all

Allow from all

Satisfy all

</Directory>

</VirtualHost>

# Supplemental configuration

#

# Load config files in the "/etc/httpd/conf.d" directory, if any.

IncludeOptional conf.d/\*.conf

Puis on configure vi /etc/httpd/sites-available/www.break-it.lan :

<VirtualHost \*:80>

ServerName www.break-it.lan

ServerAdmin username@rockylinux.org

DocumentRoot /var/www/sub-domains/www.break-it.lan/html

DirectoryIndex index.php index.htm index.html

Alias /icons/ /var/www/icons/

# ScriptAlias /cgi-bin/ /var/www/sub-domains/com.ourownwiki.www/cgi-bin/

CustomLog "/var/log/httpd/www.break-it.lan-access\_log" combined

ErrorLog "/var/log/httpd/www.break-it.lan-error\_log"

<Directory /var/www/sub-domains/www.break-it.lan/html>

Options -ExecCGI -Indexes

AllowOverride None

Order deny,allow

Deny from all

Allow from all

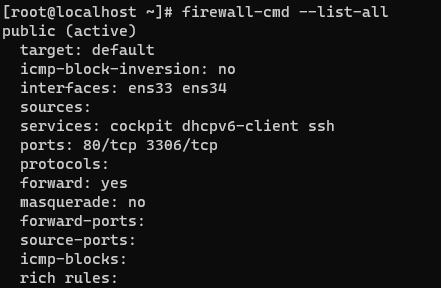
Satisfy all

</Directory>

</VirtualHost>

Puis on ouvre les port 80 pour le site et 3036 pour python. Puis on configure le sql sur le serveur web.

Puis on configure le firewall :



LA BDD :

On créait une base de données score avec une table aussi appelé score puis on va utiliser comme variable un id (un int), un nom (un varchar), un score (un int) et la date (un DATETIME) dans la base de données.

LE SITE WEB :

Pour le site, on utilise une écriture en néon pour donner un style futuriste au démineur mais aussi une animation aux menu buger. Le site web sert à savoir ce que le casse brique ? puis les règles, pour le download un bouton du jeu et du mode d’emploi. Puis la table score avec tous les score et date du record du jeux et la page contact.