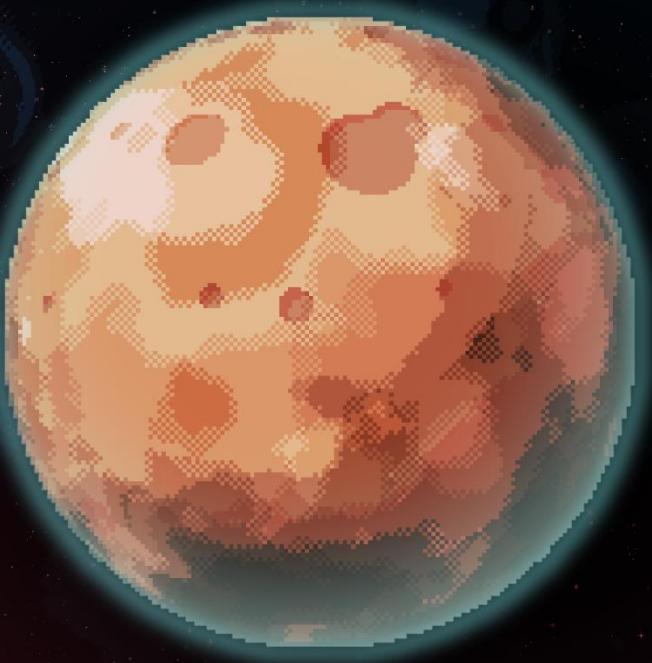


VITAE

*Vitam Explorare*



# PROIECT PENTRU OBTINEREA ATESTĂRII ÎN INFORMATICĂ



TITLUL LUCRĂRII:

VITAEEX

ELEVI: ȘTIR CEZAR, KATONA ERIC  
PROFESOR ÎNDRUMĂTOR: TOMŞA ION GELU

COLEGIUL NAȚIONAL ANDREI MUREȘANU DEJ



VITAE

~~NON SPONSORED~~

---

## CUPRINS

---

2-4 *Informații despre proiect*

5 *Realizarea proiectului*

6-15 *Programarea*

16-20 *Grafică*

21 *Cerințe hardware / software*



V | T | A | E | X

~~NON SPONSORED~~

---

## VITAEX?

---

De când eram mici ne-au plăcut jocurile video, deoarece sunt un mod foarte bun de a petrece timpul împreună. Cu cât am jucat mai multe jocuri, ne-am dat seama că acestea nu sunt doar o simplă activitate. Jocurile sunt o artă, creionată în pixeli și linii de cod. Foarte repede ne-am dat seama că ar fi super să putem să facem și noi un joc video, iar cunoștințele dobândite în timpul liceului ne-au ajutat mult.

Total a început atunci când am găsit un desen pe calculator realizat de noi cu câțiva ani în urmă. Era o navă spațială în stilul "pixel art", realizată în timpul liber. Acest desen ne-a dat ideea să începem un joc video sci-fi în stilul artistic pixel art.

Ce înseamnă VITAEX? Ei bine, este o unificare a cuvintelor "vitam" și "explorare" în limba latină și înseamnă "a explora viață". Jocul are ca scop spunerea unei povești despre diversitatea și misterele lumii. Această poveste este spusă cu ajutorul unui mic personaj extraterestru a cărui planetă este invadată. Aflându-se în pericol, el fugă în spațiu pentru a-și găsi o nouă casă într-un alt sistem solar.



# VITAE

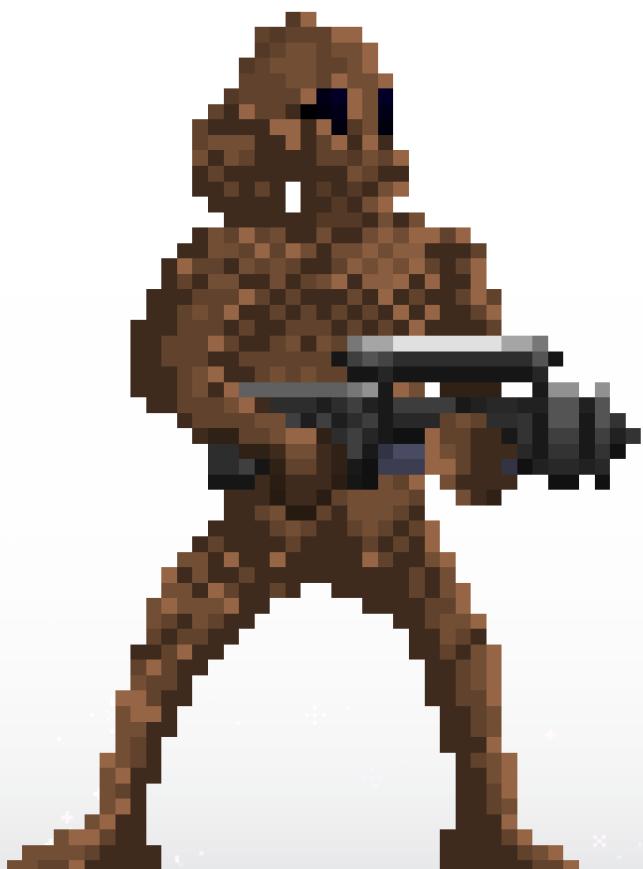
~~NON SPOILER~~

Ideea noastră a fost influențată de francize precum Star Wars, fiind niște filme nostalgitice care ne trezeau interesul pentru spațiu.

Afinitatea pentru acest domeniu a existat întotdeauna pentru noi, aşa că am încercat să ne combinăm ideile împreună și să realizăm un joc.

Personajul principal este un simplu cetățean extraterestru, făcând parte din clasa inferioară a societății de pe planeta lui. Invazia extraterestră a planetei îl determina să caute o scăpare. Dar prima dată, trebuie să găsească un mod prin care să facă acest lucru.

Planul este să ajungă la o fabrică de roboți unde știa că se confectiona un nou costum mecanizat care l-ar putea ajuta să exploreze planete extraterestre. Folosindu-se de o singură armă și o mină, acesta trece prin valuri de invadatori extraterestri și ajunge în final la fabrică. Acolo, trebuie să treacă de pericolele industriale întâlnite la fiecare pas pentru a ajunge la locul unde este confectionat robotul.





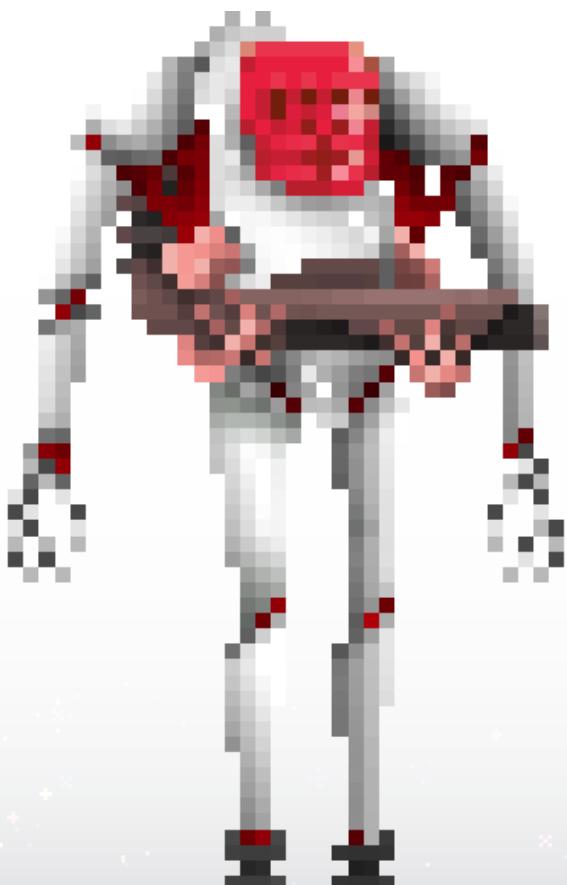
# VITAE

~~NON SPOILER~~

Povestea acestei invazii este una a trădării. Specia protagonistului a conviețuit cu specia invadatorilor timp de multe secole, încercând să se ajute cu tehnologie și resurse pentru a prospera împreună. Acest lucru i-ar fi făcut mult mai puternici în fața oricărui pericol, atât timp cât alianța dintre ei era destul de puternică.

Totuși, într-o zi oarecare, fără niciun anunț, mii de nave invadatoare începeau să zboare către planeta protagonistului. Acestea coborau la viteze hipersonice, spărgând găuri în atmosferă și începeau să tragă cu lasere în orașe. Nu cu mult timp după au apărut și navele mamă, aducând mai multe nave atacatoare și soldați. Într-un timp scurt, a avut loc o distrugere de mărimi colosale.

Motivul invadării este unul simplu. Liderul invadatorilor spune că a plănit de mult timp să preia toate resursele planetei pentru a-și face specia mai puternică. El crede că relațiile dintre cele două specii au fost degradate în urma multor dispute de-a lungul deceniilor, iar acum este timpul să eliminate total acest potențial pericol.





VITAEX

~~Mod de joc~~

---

## Realizarea

---

Proiectul VITAEX a fost realizat de-a lungul multor luni de muncă. Ambii suntem începători, aşa că am avut multe de învățat în timp ce am realizat jocul. Deși a fost foarte greu, realizarea unui joc pentru atestatul la informatică a fost un mic challenge pentru noi, iar gândul că vom putea să jucăm un joc creat de noi ne-a motivat foarte mult.

Pentru această sarcină, a trebuit să ne alegem niște programe cu care să lucrăm. Pentru partea de cod, am ales PyCharm, un program pentru programare în limbajul Python. Acest limbaj este unul dintre cele mai ușoare de învățat, aşa că a fost alegerea perfectă pentru un proiect mai complex precum VITAEX.

Pentru partea vizuală, totul a fost realizat în programul "Paint.net". Deși are un nume care aduce în gând Microsoft Paint, acest program este mult mai avansat și este mai apropiat de Photoshop. Desigur, există și alte alternative, precum GIMP, dar am ales Paint.net deoarece aveam deja experiență în acest program și muncă devinea mult mai ușoară.



# VITAE

~~NON SEDUCAT~~

Limbajul Python este folosit la scară largă în prezent, fiind foarte popular, simplu de învățat și de utilizat. Totul a pornit ca un hobby de Crăciun pentru **Guido van Rossum** în anul 1989 – fiind programator, a dorit să dezvolte un interpreter de cod pentru limbajul la care visa; simplu, intuitiv, open source, accesibil pe orice platformă. Deoarece urmarea cu drag serialul difuzat de BBC – **Monty Python's Flying Circus**, l-a numit **Python**:



### **De ce pitoni atunci?**

Până la urmă, "python" semnifică în limba engleză un piton. Chiar dacă denumirea provine de la acel serial TV, priviți puțin logo-ul oficial al limbajului de programare:

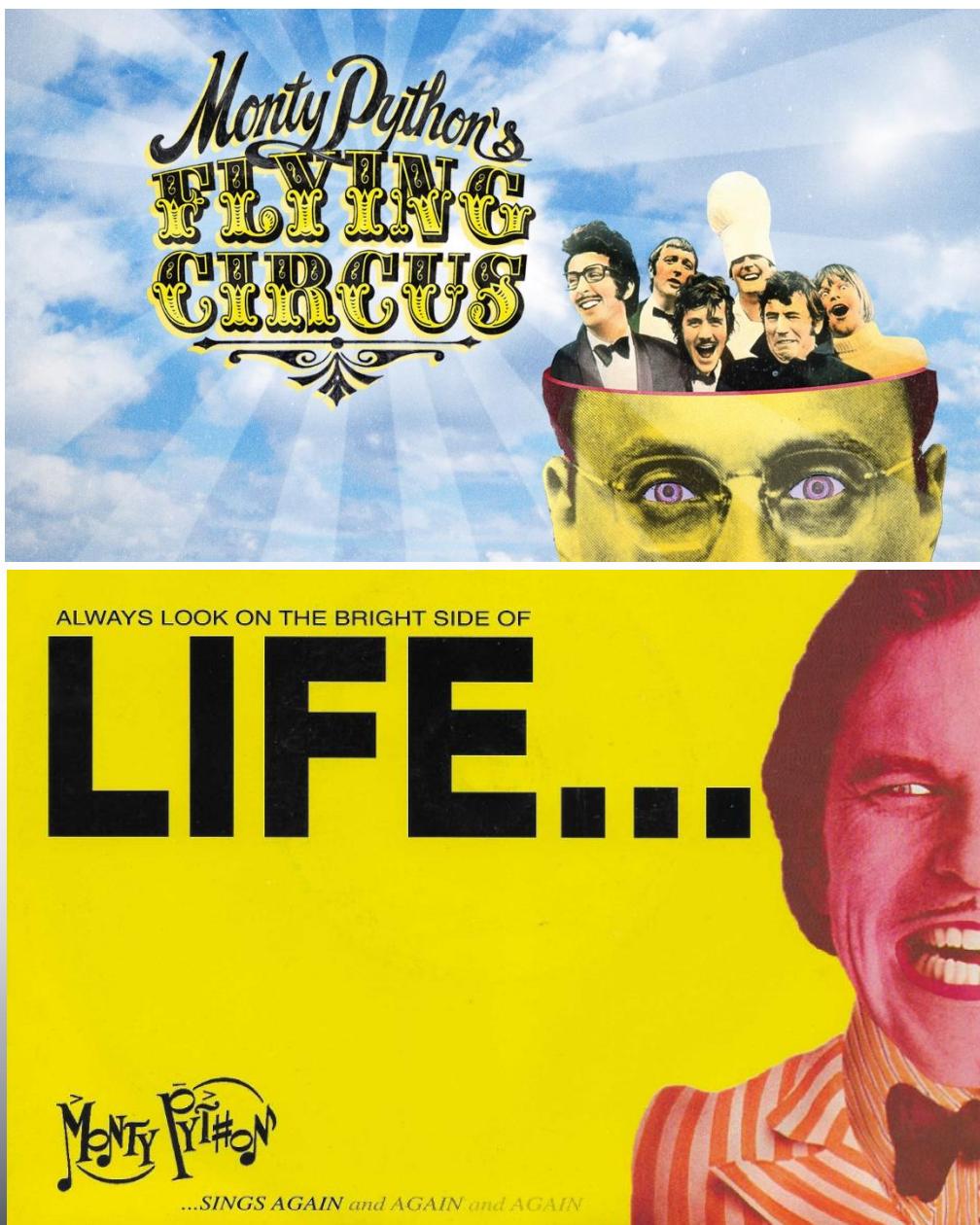




VITAE = X

~~NON SPECIFICAT~~

Deci, avem doi pitoni, clar! :) Extensia oricărui fișier Python este "**py**".  
De asemenea, priviți culorile cu care este realizat genericul serialului:





# VITAE

~~NON SOLVED~~

Cele mai importante avantaje sunt evidențiate mai jos:

- **facil de învățat / predat** – spre deosebire de alte limbaje de programare, Python se înțelege foarte repede, chiar de la început;
- **facil de folosit** – Python este simplu și minimalist, apropiat de pseudocod în limba engleză și permite ca atenția să fie focalizată pe găsirea soluției problemei și nu utilizarea limbajului în sine;
- **gratuit și open source** – este permisă redistribuirea limbajului, a codului sursă, precum și editarea / modificarea acestuia pentru a obține spre exemplu noi programe, deci poate fi și este îmbunătățit de comunitate în mod constant;
- **portabil** – limbajul Python poate fi utilizat pe o multitudine de platforme / sisteme de operare, precum: Windows, GNU/Linux, FreeBSD, Macintosh, Solaris, OS/2, PlayStation, etc.;
- **puternic orientat pe obiecte** – într-un mod foarte simplist, se pot crea cu succes aplicații folosind Programarea Orientată pe Obiecte (POO) - cam orice este un obiect, de aici și forța acestui limbaj de programare!

Limbajul Python este interpretat, adică se execută codul linie cu linie, spre deosebire de Pascal ori C/C++, unde este necesar un compilator care să genereze un fișier executabil.

Programul Python este reținut pe hard-disc în limbajul Python și nu în cod mașină, aşa cum este în cazul celor compilate.

```
>>> print("Hello World!")
Hello World!
```

```
>>>
```



VITAE

~~NON SOLUS~~

## Detalii

Un program scris într-un limbaj ce necesită un compilator este convertit la compilare din sursă în cod mașină, o succesiune de 0 și 1 (binar) care poate fi evaluată de către calculator. În linii mari, atunci când programul este executat, mediul de programare copiază programul în memoria internă și îl rulează. În cazul limbajului Python, nu este necesară compilarea programului, acesta fiind executat direct din codul său sursă. Intern, există anumite transformări a.î. sistemul respectiv să poată executa programul, însă este mult mai simplu.

Programele Python sunt cu adevărat portabile. Un program scris corect poate fi executat direct în sistemul de operare Windows, precum și în GNU/Linux, de exemplu.

Sintaxa este gândită în aşa fel încât programele Python să fie ușor de citit. Acest lucru este obținut prin folosirea de cuvinte în locul semnelor (de exemplu, and în loc de &&) și prin includerea indentării în limbaj. Astfel, în Python nu se folosesc accolade (ca în C/C++, Java), ci blocurile de cod se delimită prin indentare. Programele Python sunt, de multe ori, foarte aproape de o "implementare" echivalentă în pseudocod.

Codul este executat linie cu linie. Astfel, dacă - de exemplu - apelăm o funcție care nu există, vom primi un mesaj de eroare abia când se încearcă executarea liniei respective. Erorile de sintaxă sunt raportate însă înainte de rularea programului.

Pentru a exersa sintaxa și elementele de bază ale limbajului, vom folosi direct interpreterul Python.



VITAE

~~NON SOLUS~~

În Python nu este necesară folosirea de punct și virgulă ( ; ) la sfârșitul unei instrucțiuni. Folosim o linie pentru fiecare instrucțiune. În cazul în care dorim să scriem mai multe instrucțiuni pe aceeași linie, putem folosi ;

O caracteristică a limbajului Python este faptul că indentarea cu spații poate influența rezultatul unei secvențe de cod. Acest lucru poate părea neobișnuit pentru începători, dar este foarte logic.

Are module, clase, exceptii, tipuri dinamice și garbage collection.

## Indentarea

Indentarea în Python este mai mult decât parte a stilului de programare, este chiar parte din sintaxă.

O linie nouă termină o declarație, iar pentru a continua o declarație pe mai multe linii, se foloseste caracterul "\".

Un bloc de cod are toate liniile indentate cu același număr de spații (nu există begin și end sau {}). Instrucțiunile dintr-un bloc de cod vor fi grupate unele sub altele, pe același nivel de indentare.

Definițiile neindentate într-un program Python vor fi de obicei variabile globale, definiții de clase, proceduri, funcții sau părți ale main-ului.

După instrucțiuni nu este obligatoriu să puneti simbolul ';' precum în alte limbaje de programare, indentarea corespunzatoare fiind suficientă.

Este foarte important sa nu uitați simbolul ';' care precede o indentare.



# VITAE

~~NON SEDIMENT~~

Din punct de vedere al tipării Python folosește tipuri pentru obiecte, însă la definirea variabilelor nu trebuie precizat tipul acestora. Constraințele de tip sunt verificate la execuție (late binding), astfel încât pot apărea erori și exceptii generate de folosirea unui tip necorespunzător în atribuiri și exceptii.

Python oferă o serie de funcții predefinite (**built-in functions**) cum ar fi: `len` pentru lungimea unui obiect de tip colecție, `id` care în implementarea CPython întoarce adresa de memorie a obiectului, funcții de conversie de tip cum ar fi `str`, `int`, `float`, `bool`, etc.

Python are un număr de framework-uri care au câștigat foarte multă popularitate în ultimii ani în acest domeniu. Acestea sunt suficient de stabile și dau o viteză ridicată aplicațiilor, fiind în prezent folosite de firme precum: Spotify, Instagram, Netflix sau Uber.





VITAEX

~~NON-SECRET~~

## Mici explicații ale codului

Cel mai bun ajutor în realizarea acestui joculeț a fost librăria ‘pygame’, o bibliotecă Python folosită pentru crearea unor jocuri și aplicații multimedia.

Pentru dezvoltarea VITAEX, avem nevoie de o buclă infinită creată explicit de programator. Astfel, la fiecare pas al buclei, putem modifica starea programului în funcție de input-ul utilizatorului și de evenimentele declanșate de diferite componente ale acestuia.

*Jocul funcționează pe baza unor obiecte de tip ‘Actor’. Astfel fiecare level are proprii actori, precum backgroundul, obiectele rigide, pământul, playerul și inamicii. Fiecare actor funcționează după anumite principii, după o fizică ‘101’, pe care am creat-o.*

*Spre exemplu, fiecare planetă are propria gravitație, iar playerul se mișcă diferit, poate sări pe o distanță mai lungă sau mai scurtă în funcție de forța de atracție a planetei pe care se află.*

```
while levelController.running:
    nextLevel = levelController.nextLevel()
    if nextLevel:
        lvl = levelController.levelUp()
        levelController = LevelController(screen, lvl)
        levelController.generateEnemy()
        levelController.generateGuardian()
        levelController.generatePress()
        levelController.generateCable()
        levelController.generatePipe()
        levelController.generateLaser()
        levelController.generateGate()
        levelController.updateActorsList()
        levelController.updateObjectsList()
        levelController.updateRigidBodies()
        nextLevel = False

    clock.tick(60)

    levelController.drawEnvironment()

    if not levelController.player.isShot:
        levelController.tickGame()
    else:
        gameOver()
        pygame.display.update()

    pygame.quit()
```



# VITAE

~~MAP SECRET~~

Fiecare actor conține atribuții specifice, precum și niște stări specifice:

```
self.velocity = 7
self.gravityForce = GFORCE
self.spritesheet = resource.spritesheet
self.isFalling = False
self.isIdle = True
self.isRight = True
self.isLeft = False
self.animationList = []
self.animationListFlip = 0
self.animationSteps = resource.animationFrames
self.action = 0
self.animationCooldown = 90
self.frame = 0
self.stepCounter = 0
self.scroll = 0
self.actionChanged = False
self.topOffset = 0
```

Fiecare level are proprii săi actori, care formează o scenă reprezentativă. Primul nivel conține un background, format din mai multe layere, care se mișcă la viteze diferite pentru a se creea un efect de parallax, inamici, și bineînțeles, playerul.

Nivelul al doilea, este un nivel de tip 'platformer' ce conține mai multe obstacole, precum o țeavă din care ieșe un abur fierbinte, un laser, o presă, și un guardian ce stă de pază în fața unei uși. Fiecare obstacol este un obiect diferit și are clasa sa proprie, cu funcții proprii.



# VITAE

~~MAP SECRET~~

```
class Press:  
    ▲ paunescudanutz +1  
    def __init__(self, pressPosX, pressPosYOffset, resourceProvider):  
        self.up = True  
        self.offset = pressPosYOffset  
  
        self.pressUp = Actor((pressPosX, -2 - pressPosYOffset),  
                            2,  
                            resourceProvider.getResource("pressUp"),  
                            None)  
  
        self.pressDown = Actor((pressPosX, (272 * 2) - 2),  
                             2,  
                             resourceProvider.getResource("pressDown"),  
                             None)  
  
    ▲ CaesarHD  
    def drawActor(self, screen):  
        self.pressUp.drawActor(screen)  
        self.pressDown.drawActor(screen)  
  
    ▲ CaesarHD  
    def pressPlayer(self, player):  
        return abs(player.getCollisionBox().top - self.pressUp.getCollisionBox().bottom) > 5  
  
    ▲ paunescudanutz  
    def presses(self):  
        if self.pressUp.getCollisionBox().bottom >= self.pressDown.getCollisionBox().top:  
            self.up = True  
        elif self.pressUp.getCollisionBox().bottom < 170:  
            self.up = False  
  
    ▲ CaesarHD +1  
    def playerPressed(self, player):  
        collisionHeight = self.pressDown.bounds.top - self.pressUp.bounds.bottom  
        collisionArea = Rect(self.pressUp.bounds.x + 10, self.pressUp.bounds.bottom, self.pressUp.bounds.width - 20,  
                            collisionHeight)
```

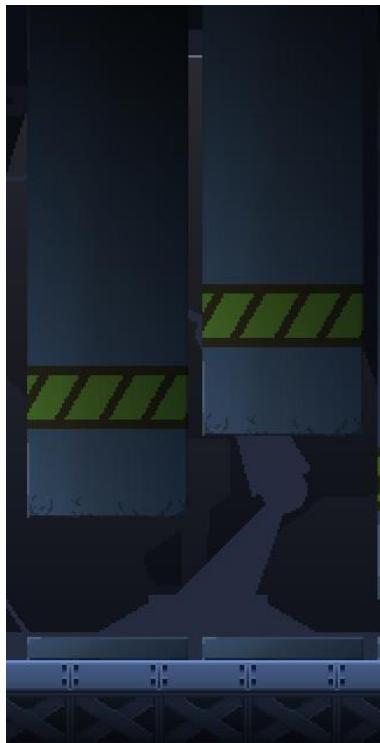
Acesta este un fragment de cod din joc, reprezentând clasa 'press'. Aceasta primește din exterior 3 parametrii: o poziție pe axa X, numită "pressPosX", un offset pentru axa Y, deoarece presele vor fi una lângă alta și vor începe din poziții diferite pentru o dificultate ridicată a jocului, și o resursă. În funcție se creează 2 obiecte ce formează o presă. Partea de sus, care va fi în continuă mișcare, și partea de jos pe care va ateriza.

În clasă am declarat mai multe funcții care alcătuiesc mecanismul de funcționare a presei. Astfel, se remarcă o funcție care desenează pe ecran obiectele care formează presă, o funcție care returnează True dacă playerul se află în aria determinată de spațiul dintre cele două componente ale presei.



# VITAE

~~MAP SPIDER~~



Toți actorii din joc conțin un spritesheet cu un anumit număr de frame-uri, pentru a se realizează animația. Astfel, când declarăm un actor, trebuie să precizăm numărul frame-urilor din fiecare animație:

```
self.playerAnimationFrames = [4, 6, 3, 1, 2, 4, 6, 3, 1, 2, 3, 6, 2, 3, 1, 6, 7, 7]
```

Pentru fiecare animație, se iterează prin fiecare frame și se adaugă fiecare acțiune a actorului într-un vector. Astfel, se formează un vector ce conține animații, iar fiecare animație la rândul ei este un vector ce conține frame-uri.

Fiecare actor din joc are gravitație și este tras în jos până se intersectează cu un alt actor. Astfel, actorii au un collision box cu care se realizează interacțiunea.

```
for animation in self.animationSteps:  
    tempImageList = []  
    for _ in range(animation):  
        tempImageList.append(self.spritesheet.getImageByIndex(self.stepCounter, self.scale))  
        self.stepCounter += 1  
    self.animationList.append(tempImageList)
```



VITAE

~~NON SPONSORED~~

---

## Grafica

---



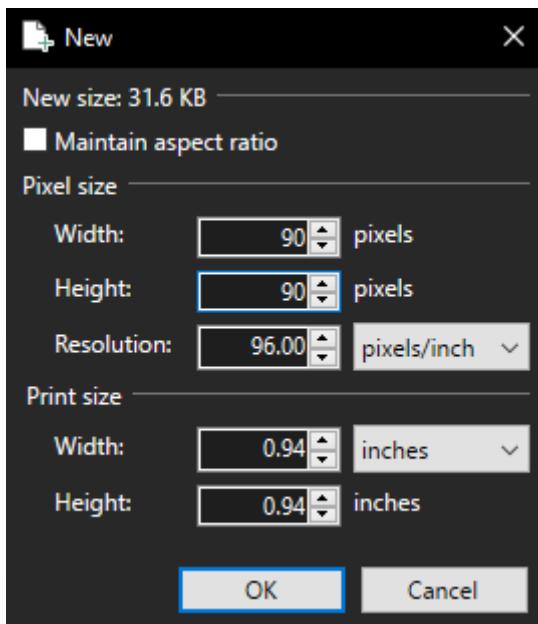
paint.net™

Grafica jocului presupune tot ceea ce vede jucătorul. Cum arată personajele, background-urile, nivelul în sine, animațiile etc. Toate acestea au fost realizate în programul "Paint.net", un program gratuit pentru editare foto.

Programul are o istorie interesantă: a fost scris în intermediul unui proiect de către un student numit Rick Brewster la un curs de Știința Calculatoarelor în anul 2004, la Washington State University. Prima versiune a avut 36,000 de linii de cod și a fost scrisă în cincisprezece săptămâni. După puțin timp, a avut peste două milioane de instalări, devenind un program preferat pentru editare foto. Este ușor de învățat, gratuit și primește actualizări chiar și în ziua de azi.



Deoarece jocul este în stilul pixel art, grafica jocului trebuie să fie creată la o rezoluție foarte mică. Acest lucru este simplu de făcut: atunci când deschidem o filă nouă, putem să alegem orice rezoluție dorim:



După crearea unei file, suntem prezenți cu o pagină albă. Aici putem să începem desenul propriu-zis, folosind o mulțime de unelte oferite de programul Paint.net: **Paintbrush**, **Pencil**, **Fill Bucket**, **Gradient**, **Eraser**, **Recolor**, **Line/Curve** și multe altele. Cea mai folosită este Paintbrush, fiind una dintre bază cu care se realizează desenul. Avem opțiunea de a alege orice culoare folosind sistemul RGB (Red Green Blue).





# VITAE

~~NON SPOILER~~

Animațiile din joc sunt realizate prin metod frame-by-frame. Această metodă presupune desenarea mai multor poze și rularea lor la o viteză rapidă pentru a crea o animație. Acesta este un mod clasic de animare, fiind modul în care au fost realizate majoritatea desenelor animate vechi și chiar și primul video din istorie, cu un cal care alerga. Programul Paint.net are și funcția de 'layere', ceea ce este extrem de important pentru desenarea în sine. În esență, ne permite să desenăm mai multe lucruri în același proiect, iar la final, sunt suprapuse pentru a crea desenul. Layerele oferă un mod de lucru eficient, deoarece este o problemă foarte mare când desenăm o componentă a unui personaj, iar aceasta se suprapune cu altă componentă, deoarece trebuie ștearsă o parte din desen. Cu layere, putem să desenăm mâinile, capul, hainele, arma și alte detalii complet separate una față de celalătă.

Un exemplu de animație frame-by-frame: Alergare



După ce toate frame-urile sunt gata, folosim un website numit "[Imgflip](#)" unde putem pune toate pozele pentru a viziona animația sub forma unui fișier '.GIF'.

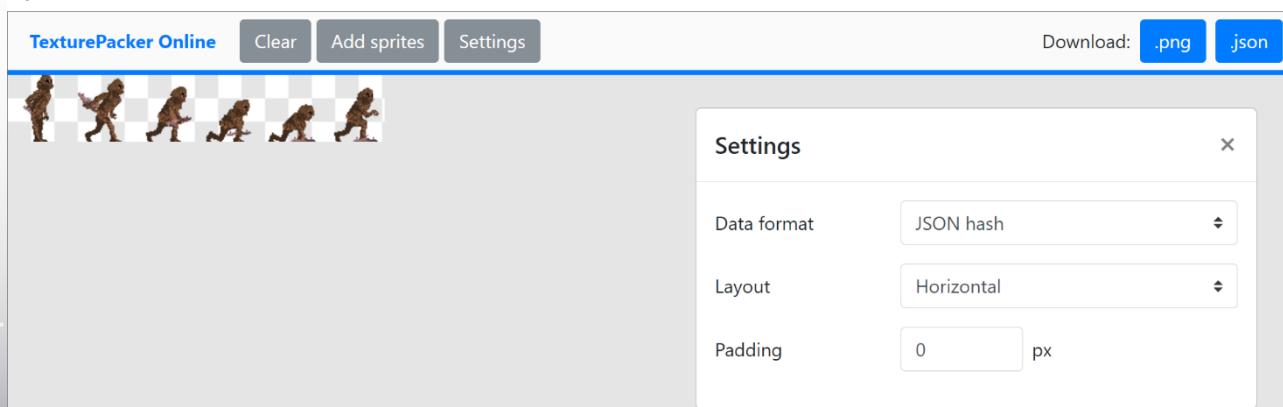
Acest lucru este foarte folositor deoarece putem vedea cum arată animația înainte de includerea ei în joc, iar erorile pot fi reparate.



După ce animația este gata și toate frame-urile au fost completate, în format '.PNG', este timpul ca ele să fie incluse în joc. Acest lucru este realizat prin intermediul unui "spritesheet", o singură poză care este alcătuită dintr-o însiruire de frame-uri fără spațiu între ele, reprezentând animația sub forma unei poze. Pentru realizarea unui spritesheet, am folosit un website numit "[Free Sprite Sheet Packer](#)". Aici putem încarca fiecare frame, iar site-ul le va pune într-un spritesheet pentru noi. Acest lucru este foarte util, deoarece minimalizează erorile făcute de noi la îmbinarea pozelor.

Folosirea unui spritesheet este benefică și pentru rularea jocului, deoarece cu cât o animație este mai complexă, cu atât va avea mai multe frame-uri. În schimb, cu un spritesheet, animația este reprezentată printr-un singur fișier în loc de o mulțime de fișiere.

Spre exemplu, în total, jocul VITAE are aproximativ 67 de fișiere numai pentru animațiile personajului principal. Prin utilizarea unor spritesheet-uri, acest număr este redus la aproximativ 8 fișiere.





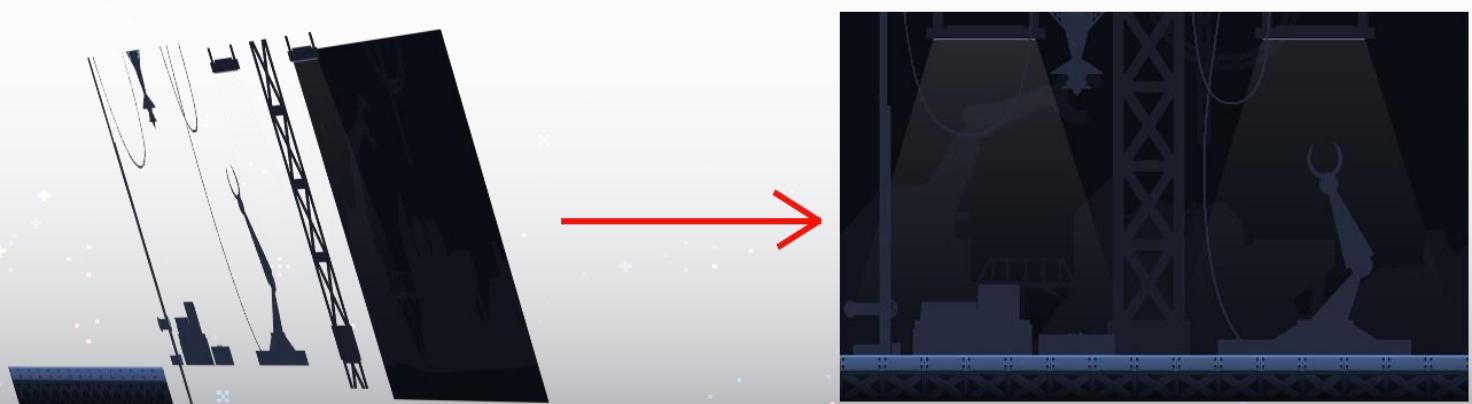
# VITAE

~~MAP SECRET~~

Un efect utilizat foarte des în jocurile pixel art este efectul "parallax". Prin intermediul acestui efect, lumea din joc prinde viață și pare ca și cum ar fi mult mai mare decât este defapt. Lucrurile îndepărtate se mișcă încet, iar lucrurile apropiate de jucător se mișcă mai repede, având un efect aproape 3D.

Pentru realizarea efectului avem nevoie de un lucru amintit atunci când am prezentat programul Paint.net: [layer!](#)

În acest caz, fiecare layer va reprezenta o parte din background, clasificate pe distanțe: layerul cel mai din față este cel apropiat, apoi cel mediu, cel îndepărtat și aşa mai departe. Pentru desenarea background-ului, trebuie să avem grijă ca acesta să fie confectionat pentru "loop". Asta înseamnă că dacă luăm background-ul și îl repetăm în continuarea sa, va avea loc o tranziție 'seamless'. În cod, background-ul este defapt format dintr-o singură imagine care se tot repetă. Același principiu are loc și la efectul de parallax, doar că viteza cu care se mișcă poza va fi diferită în funcție de layer.





VITAE

~~NON SPONSORED~~

Un alt website folosit este "[WeTransfer](#)". Este un site gratuit care permite transferul fișierelor de orice tip prin intermediul unui link care este copiat și trimis destinatarului. Acesta ne-a permis să trimitem fișiere unul de la celălalt fără pierderi de calitate sau coruperea fișierelor și a fost extrem de util!

### *Cerintele minime hardware și software:*

OS: Windows 7, 8, 8.1, 10 x64 sau x32

Procesor: Intel Pentium E2180 (2\*2000) sau echivalent

Memorie: 500 MB RAM

Grafică: GeForce 7600 GT (256 MB) sau echivalent

Spațiu: ~500 MB

