Proyecto Alien

Integrantes: Alan Pérez, Micaela Peralta, Nicolás Ojeda, UNLa, 29 de septiembre 3901, Buenos Aires, Argentina.

Resumen

En este proyecto vamos a tratar el armado de un juego en el lenguaje de programación Python, desde la planificación, el diseño y análisis, programación, documentación y testeo de software.

En este breve resumen, trataremos acerca de la historia del juego:

A millones de años luz de la tierra, se encuentra nuestro personaje Alien, quien tuvo dificultades con su nave espacial en un vuelo de emergencia y aterrizó en un planeta desconocido. Su misión en el primer nivel es armarse de valor y esquivar diversos objetos que caen desde las alturas, para poder recoger las piezas perdidas de su nave y poder emprender el viaje de vuelta a su hogar. Cuando lo consigue, ya nuevamente en su nave, debe evitar los meteoritos espaciales que complican su vuelta y volar hacia su planeta.

Palabras clave:

-Programación, Python, juego, desarrollo, producción, software –

Abstract

In this project we are going to discuss the creation of a game in the Python programming language, from planning, design and analysis, programming, documentation and software testing.

In this brief summary, we will discuss the history of the game:

A million light years from Earth, we find our character Alien, who struggled with his spacecraft on an emergency flight and landed on an unknown planet. His mission on the first level is to arm himself with courage and dodge various objects that fall from the heights, in order to collect the lost pieces of his ship and be able to embark the trip back to his home. When you get it, and again in your ship, you must avoid the space meteorites that complicate your return and fly to your planet.

Key words: -Programming, Python, games, development, software, production-

Introducción

El proyecto se encuentra dividido y pensado en 4 partes:

- Documentación del producto
- Documentación técnica
- Código
- RRHH

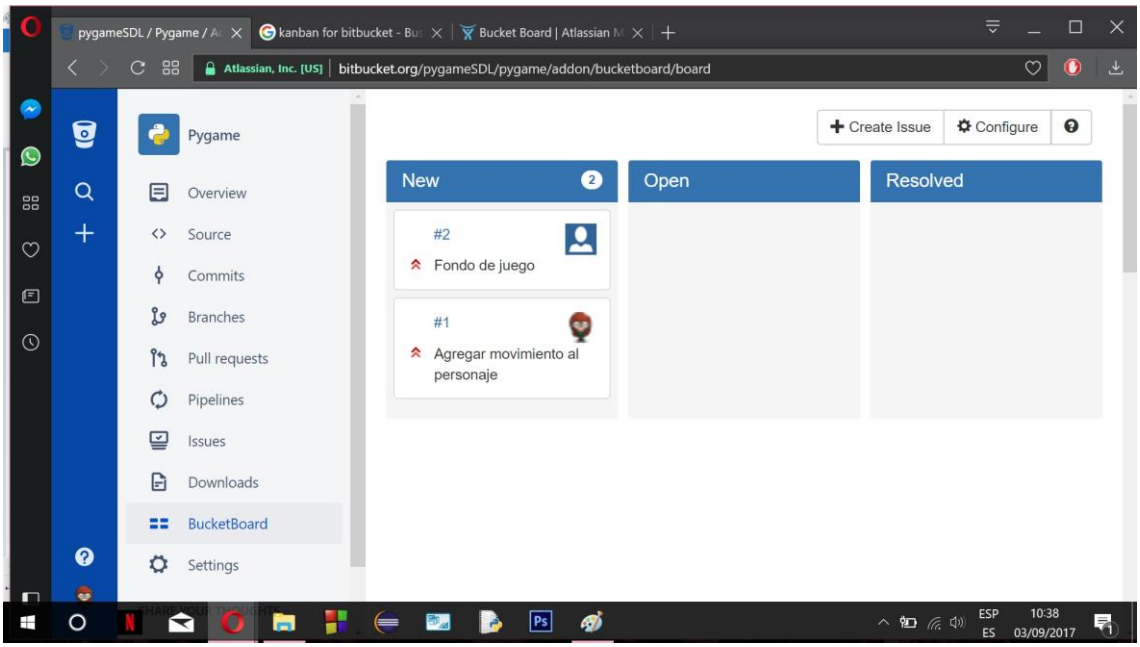
En cada una de ellas se encuentra ordenado, para una mejor organización, las partes del juego. Pasando desde la guía de usuario, imágenes, sonidos, música y el código completo del juego.

En cuanto a la programación, utilizamos un control de versiones para poder ahorrar tiempo, trabajar en equipo simultáneamente y, en caso de errores, poder volver a una versión estable del código. Para todo esto usamos el software controlador de versiones Git y el cliente SourceTree.

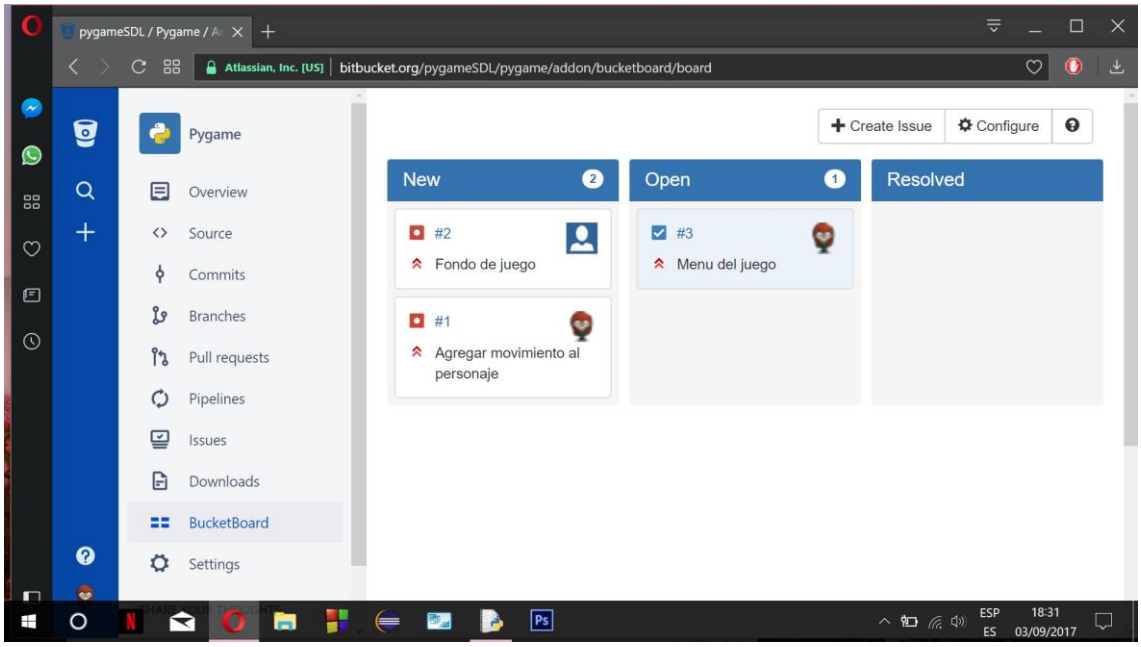
Para tener un control y poder planificar correctamente el juego, diariamente realizamos una serie de minutas en las cuales comentábamos en grupo ideas y aportes generales. A medida que empezamos la programación y sólo por control, realizamos reuniones de avances, en las cuales anotamos las cosas que modificamos del código, que día y la hora correspondiente.

Como ayuda para nosotros, utilizamos una de las metodologías ágiles de organización de tareas: Kanban y a continuación dejaremos algunas capturas de pantalla en el medio de la producción del juego:

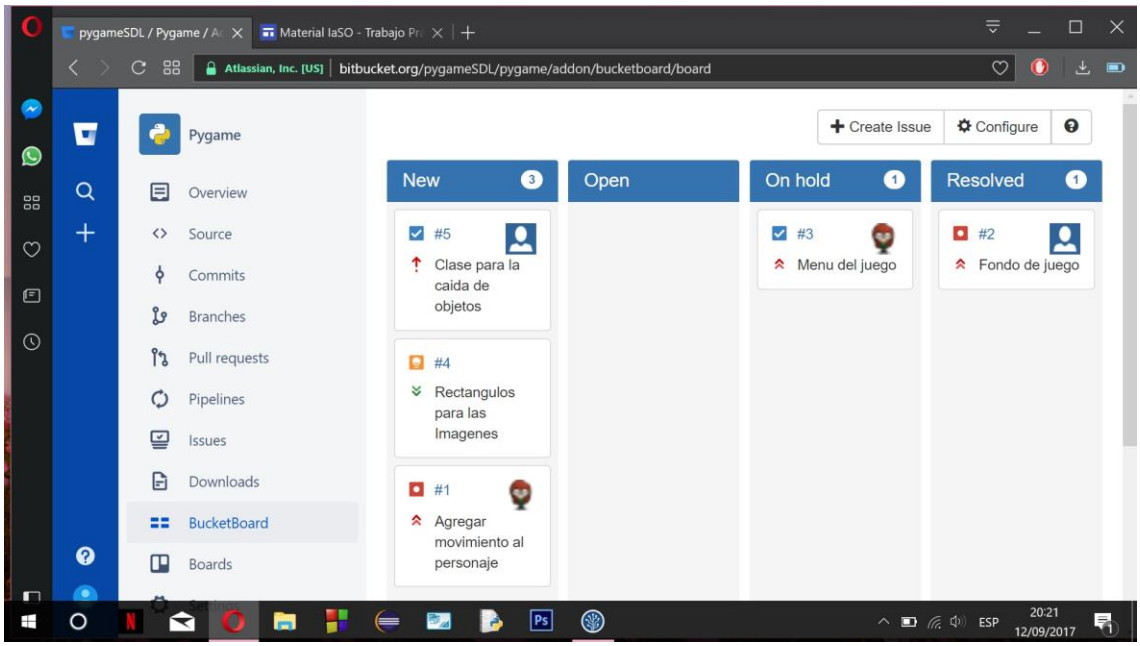
1)


pygameSDL / Pygame /
kanban for bitbucket -
Bucket Board | Atlassian
Atlassian, Inc. [US]
bitbucket.org/pygameSDL/pygame/addon/bucketboard/board
Pygame
Overview
Source
Commits
Branches
Pull requests
Pipelines
Issues
Downloads
BucketBoard
Settings
Create Issue
Configure
New
2
Open
Resolved
#2
Fondo de juego
#1
Agregar movimiento al personaje
ESP
ES
10:38
03/09/2017

2)


pygameSDL / Pygame /
Atlassian, Inc. [US]
bitbucket.org/pygameSDL/pygame/addon/bucketboard/board
Pygame
Overview
Source
Commits
Branches
Pull requests
Pipelines
Issues
Downloads
BucketBoard
Settings
Create Issue
Configure
New
2
Open
1
Resolved
#2
Fondo de juego
#3
Menu del juego
#1
Agregar movimiento al personaje
ESP
ES
18:31
03/09/2017

3)

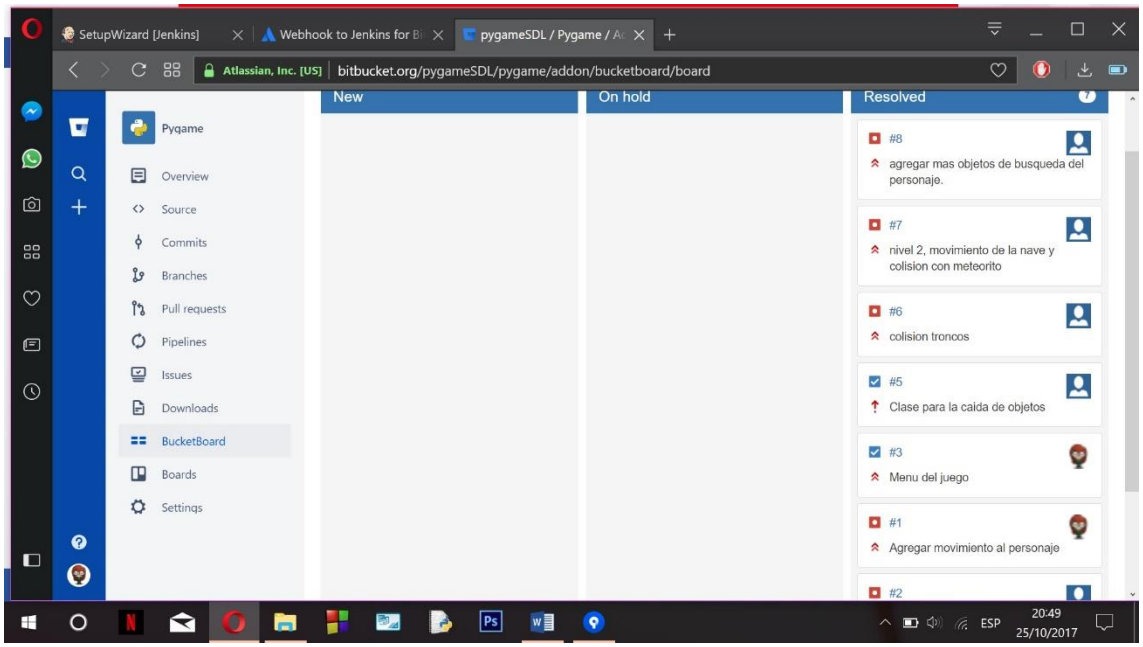


En estas imágenes podemos observar cómo se dividen y asignan las diversas tareas para la producción del juego:

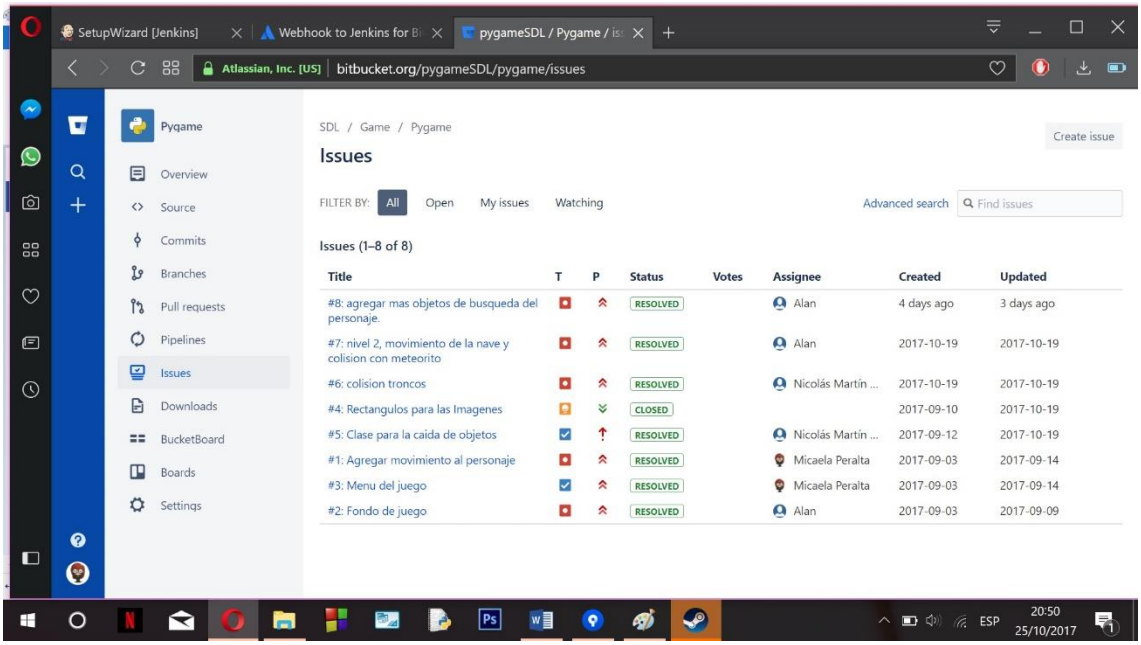
En la imagen 1) (3/9) se ponen en el tablero las tareas o problemas a resolver.

La imagen 2) muestra que la tarea ya fue abierta y está en proceso de ser resuelta, como se muestra en la imagen número 3), que la tarea pasa de “Open” a “On hold” y luego, una vez finalizada, pasa a la columna “Resolved”.

4)



5)



En las imágenes 4) y 5) observamos como todas las tareas que fueron asignadas se llevaron a cabo hasta su resolución. La imagen 5) muestra desde otro punto las tareas, con un sistema propio del control de versiones “Bitbucket”.

Introduction

The project is divided into four parts:

- *Product documentation*
- *Technical documentation*
- *Code*
- *HR*

In each one of them is arranged, for a better organization, the parts of the game. Passing from the user guide, images, sounds, music and the complete game code.

As for programming, we use a version control to save time, work in teams simultaneously and, in case of errors, to be able to return to a stable version of the code. For all this we use the Git version driver software and the SourceTree client.

To have a control and to be able to plan the game correctly, daily we made a series of minutes in which we discussed in group ideas and general contributions. As we begin programming and only by control, we hold progress meetings, in which we note the things we modify the code, which day and time.

As an aid to us, we use one of the agile methodologies of task organization: Kanban and then we leave some screenshots in the middle of the production of the game.

In these images we can see how the various tasks for the production of the game are divided and assigned:

In figure 1) (3/9) the tasks or problems to be solved are put on the board.

Image 2) shows that the task has already been opened and is in the process of being solved, as shown in image number 3), that the task changes from "Open" to "On hold" and then, to the "Resolved" column.

In pictures 4) and 5) we observed how all the tasks that were assigned were carried out until their resolution. The image 5) shows from another point the tasks, with an own system of the control of versions "Bitbucket".

Estado de la cuestión

Gracias al uso de las diferentes metodologías ágiles, es fácil (casi siempre) notar los diversos “bugs” o problemas que pueda presentar el programa en cuestión y para mayor rapidez de resolución de conflictos, asignamos la tarea a uno de los integrantes para que proceda mediante Kanban. Si por algún motivo le surgiera un problema, otro integrante deberá solucionar y asignarse el conflicto lo antes posible.

Para la detección de problemas, existen herramientas de integración continua que permiten detectar fallos lo más rápido posible mediante pruebas métricas y de calidad, compilando el código desde el repositorio central del control de versiones. Esto se hace periódicamente, tantas veces como lo configuremos y lo que permite es detectar errores de compilación o pequeños “bugs” que puedan surgir y así solucionarlo lo más rápido posible para una mejor entrega del producto final.

Consideraciones técnicas

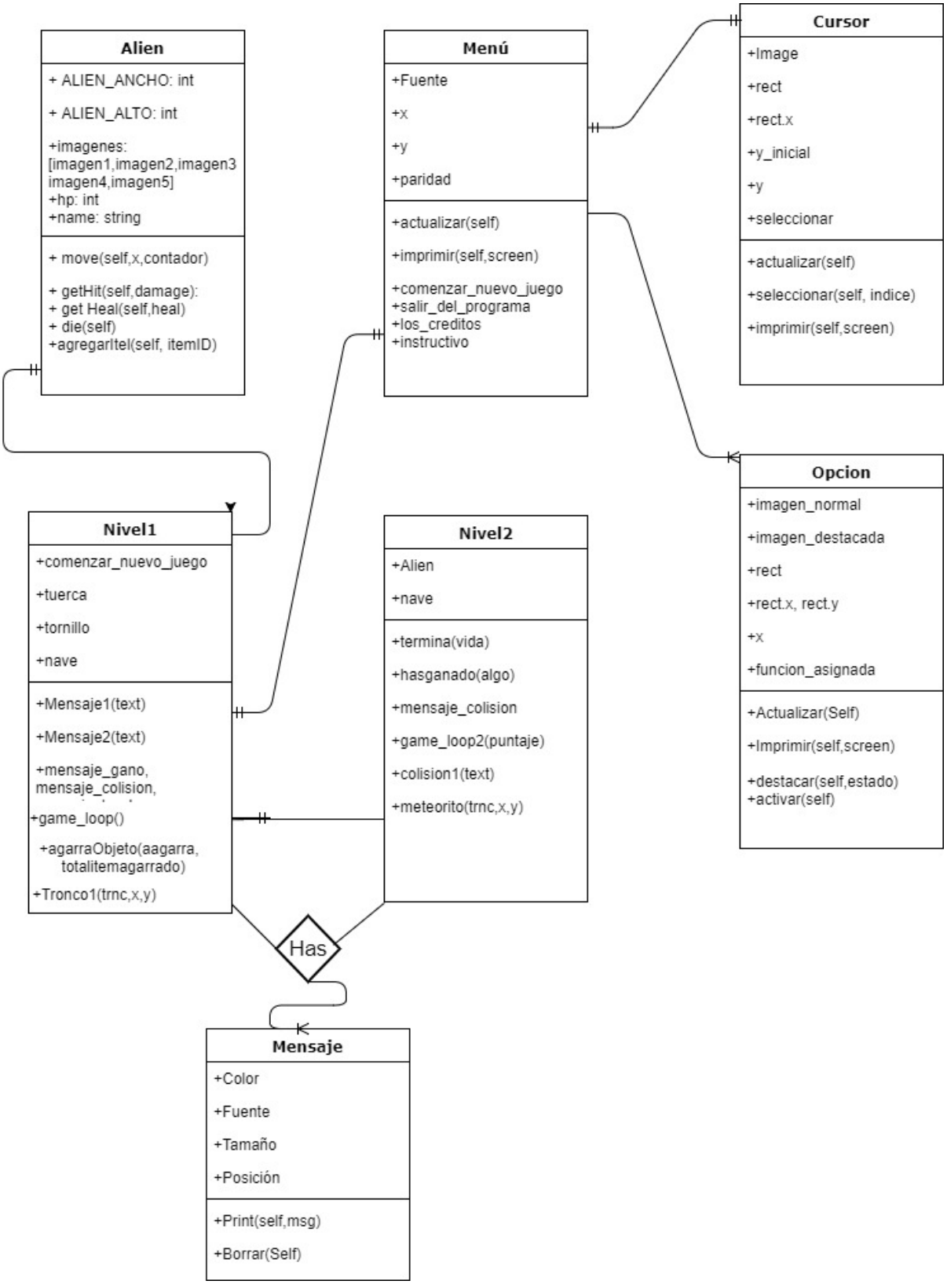
Para comenzar con las especificaciones del juego descripto anteriormente, podemos decir que usamos un paradigma de programación orientado a objetos, lo que nos permite definir e identificar a los objetos frente a otros. Está dividido en clases, las cuales se encargan de definir las propiedades y comportamiento de un objeto concreto. El objeto es una instancia de la clase, con los comportamientos y propiedades asignados en la misma. Principalmente para la programación del juego, utilizamos Pygame, que es una librería propiamente del lenguaje de programación Python, de fuente abierta y gratuita. Pygame resulta muy versátil, ya que se ejecuta en casi cualquier plataforma y sistema operativo y trae consigo diversas herramientas que son de gran utilidad y hacen más fácil el desarrollo del juego.

En la siguiente página podemos observar un diagrama de clases, hecho por uno de los integrantes, en el cual se detallan las clases, atributos y métodos usados en la programación del juego. No obstante, quedan afuera algunas partes del juego que no están detalladas en el diagrama.

Una explicación para este diagrama comenzaría por la clase “Menu”, la cual a su vez se relaciona con las clases “Opciones”(1-muchos) y “Cursor”(1-1). A su vez, “Menu” se relaciona con “Nivel 1” (1-1) y éste, a su vez, con “Nivel 2” en una relación de mandato (Si no se pasa el nivel 1, no existe nivel 2). Ambos niveles tienen mensajes (1-muchos).

Por otra parte, la clase “Alien” se encuentra ligada al “Nivel 1” y por las relaciones de éste, consecuentemente al “nivel 2”.

A continuación el diagrama de clases:


Alien
+ ALIEN_ANCHO: int
+ ALIEN_ALTO: int
+imagenes: [imagen1,imagen2,imagen3 imagen4,imagen5]
+hp: int
+name: string
+ move(self,x,contador)
+ getHit(self,damage):
+ get Heal(self,heal)
+ die(self)
+agregarItel(self, itemID)
Menú
+Fuente
+x
+y
+paridad
+actualizar(self)
+imprimir(self,screen)
+comenzar_nuevo_juego
+salir_del_programa
+los_creditos
+instructivo
Cursor
+Image
+rect
+rect.x
+y_inicial
+y
+seleccionar
+actualizar(self)
+seleccionar(self, indice)
+imprimir(self,screen)
Opcion
+imagen_normal
+imagen_destacada
+rect
+rect.x, rect.y
+x
+funcion_asignada
+Actualizar(Self)
+Imprimir(self,screen)
+destacar(self,estado)
+activar(self)
Nivel1
+comenzar_nuevo_juego
+tuerca
+tornillo
+nave
+Mensaje1(text)
+Mensaje2(text)
+mensaje_gano, mensaje_colision,
+game_loop()
+agarraObjeto(aagarra, totalitemagarrado)
+Tronco1(trnc,x,y)
Nivel2
+Alien
+nave
+termina(vida)
+hasganado(algo)
+mensaje_colision
+game_loop2(puntaje)
+colision1(text)
+meteorito(trnc,x,y)
Has
Mensaje
+Color
+Fuente
+Tamaño
+Posición
+Print(self,msg)
+Borrar(Self)

Futura línea de investigación

Seguiremos desarrollando y mejorando el trabajo con las diferentes herramientas que están al alcance de cualquier programador. Las metodologías agiles e integración continua son en las que reforzaremos y trabajaremos para mejorar su uso, dado que pudimos observar que realmente son benéficas para el grupo de desarrollo porque permiten ahorrar muchos tipos de errores comunes y simples que retrasan el trabajo. Mediante esta práctica de desarrollo de software como es la *integración continua*, se combinan cambios en el código del repositorio central del sistema de control de versiones (en este caso Bitbucket) periódicamente y se ejecutan versiones y pruebas automáticas, detectando al instante cualquier error funcional o de integración y mejorando la calidad del software.

Conclusiones

Luego de haber finalizado la producción del juego, podemos concluir que una parte muy importante del ambiente de desarrollo fueron las metodologías ágiles y el controlado de versiones. Sin el último hubiese llevado mucho más tiempo y trabajo darnos cuenta de errores y, a la vez, solucionarlos sin poder volver a una versión estable sería laborioso. Sabemos que trabajar en equipo no siempre resulta cómodo para todos, pero este tipo de sistemas hace que el proceso sea más eficiente y logra una mayor equidad a la hora de realizar tareas.

Otro punto es la correcta documentación del programa, la cual ayuda a la hora de analizar y agregar contenido y además da un mejor entendimiento al usuario.

Referencias:

https://www.iconfinder.com

https://gamethemesongs.com/

https://es.stackoverflow.com

https://www.pythonmania.net

https://pythonprogramming.net

https://www.python.org/

https://www.pygame.org/wiki/about

Agradecimientos

A Federico Perez, por la creación de la imagen del personaje principal del juego. A Attlassian, por prestar el servicio gratuito de Bitbucket. A Stackoverflow por resolver dudas acerca de Python.