**DESARROLLO DEL JUEGO CUBO DE RUBIK EN PYTHON**

**Huaman Ancco Yuver**

**Quintana Olarte Elizabeth**

***UNAJMA. Universidad nacional José María Arguedas***

**Abstrac**

El cubo de bolsillo (también conocido como Minicubo, Mini Rubik o Cubo de hielo) es el equivalente de un Cubo de Rubik pero de dimensión 2x2x2. Aunque mecánicamente es más complejo que el Cubo de Rubik, su resolución es mucho más sencilla, ya que es análoga a reordenar únicamente los vértices de un cubo de Rubik estándar 3x3x3.

El principal objetivo de este proyecto es el desarrollo del juego cubo de rubik 2x2, utilizando el lenguaje de programación Python y haciendo uso de **las librerías numpy y matplotlib** para así resolver mediante una serie de algoritmos, el juego Cubo de Rubik tiene una **interfaz amigable y sencilla** lo que hace posible una fácil comprensión por parte del usuario

**Palabras Clave**

**Las librerías numpy y matplotlib, interfaz amigable y sencilla.**

**Introducción**

El cubo de rubik ha sido objeto de análisis desde sus orígenes debido a sus características. Un problema con un espacio de estados inmenso, siempre se han buscado métodos y maneras de resolverlo.

Actualmente existen varios métodos para que las personas puedan resolver el cubo de rubik rápidamente, pero también se buscan maneras en que un ordenador pueda resolver el cubo en un tiempo razonable, o en un número de movimientos mínimo.

Se ha podido demostrar que el número de movimientos máximo para resolver cualquier cubo es 20. Esto se ha sabido resolviendo todos los estados del cubo de rubik con una búsqueda que garantizada la admisibilidad.

Actualmente se pueden encontrar aplicaciones que resuelven el cubo en poco tiempo. Algunas son admisibles, pero o tardan mucho tiempo, u ocupan mucho espacio de disco duro porque necesitan instalar bases de datos de patrones muy grandes.

En este proyecto se busca una solución rápida, sin que ello signifique consumir mucho disco duro. Nos centraremos en la optimización en tiempo.

El juego consistirá en llegar a armar las seis caras del mismo color, está desarrollado en lenguaje de programación Python y se hace uso de matrices.

**¿Cómo es el cubo de Rubik?**

En el cubo de Rubik clásico cada cara tiene nueve cuadrados de un color determinado. El cubo de Rubik está resuelto cuando cada cara es de un mismo color.

En el cubo mágico, cada una de sus caras está dividida en 9 cuadrados. Es decir, este cubo está formado por un total de 27 piezas ensambladas entre sí y unidas a una pieza central que permanece oculta.

En el cubo de Rubik hay seis piezas centrales que tienen un color cada una y siempre mantienen su posición. En el cubo de origen, el color blanco era el opuesto al amarillo, el rojo al color naranja y el verde al color azul.

También está formado por doce piezas de arista que son las que se encuentran en los bordes y que tienen dos colores diferentes. También hay ocho piezas vértice que son las que se encuentran en las esquinas y que tienen tres colores cada una.

Los primeros cuadrados, los centrales, están unidos a una pieza central mediante tornillos y se pueden girar 360 grados. Al moverlas, estas piezas arrastran a las que tienen a su alrededor.

Después del éxito del cubo de Rubik aparecieron en el mercado otras variantes del cubo mágico como La Venganza de Rubik (4x4x4), el cubo de bolsillo (2x2x2) o el cubo del profesor (5x5x5).

**Métodos**

**Antecedente:** Seguramente más de una vez te frustraste tratando de que cada una de las caras del cubo Rubik quedase de un solo color. Es que este puzle ha perdurado en el tiempo y aún hoy, en donde todo es tecnología, sigue atrayendo a gente de todas las edades.

Creado en 1974, el juego fue un invento de Ernő Rubik, profesor de arquitectura en Budapest, Hungría. Ernő estaba obsesionado con las figuras geométricas y la construcción de las mismas en 3D, llevaba tiempo intentando que éstas se pudieran mover internamente sin romperse.

Frustrado, el profesor salió a caminar por las orillas del río Danubio y al ver las piedras redondas que se alinean en la orilla, tuvo una revelación para dar con la solución a sus problemas.

Erno se dio cuenta que, si cada uno de los bloques giraba en torno a un centro redondeado, estos podrían moverse libremente, sin romperse ni perder la forma de cubo. Inmediatamente se puso a trabajar en el primer prototipo, compuestos de 21 diminutos cubos. Así nació el cubo Rubik, que en un principio llevaba pegatinas adheridas a cada uno de los lados de cada pequeño cubo. Así al pasar de los años se creó nuevas versiones del cubo 2x2, 3x3, 4x4.

A continuación presentamos un modelo muy útil para emplearse cuando existe la necesidad de desarrollar software como instrumento de medición en un estudio. En este documento nos enfocaremos a desarrollar un esquema por etapas a partir del Modelo de Investigación metodología de desarrollo programemg extremen (xp).

**Metodología extremen programeng (xp)**

Es un marco de trabajo para la gestión y desarrollo de software bajo un entorno de desarrollo ágil de software. La simplicidad es la base de la programación extrema. Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento.

Para mantener la simplicidad es necesaria la refactorización del código, ésta es la manera de mantener el código simple a medida que crece. También se aplica la simplicidad en la documentación, de esta manera el código debe comentarse en su justa medida, intentando eso sí que el código esté autodocumentado. Para ello se deben elegir adecuadamente los nombres de las variables, métodos y clases. Los nombres largos no decrementan la eficiencia del código ni el tiempo de desarrollo gracias a las herramientas de autocompletado y refactorización que existen actualmente. Aplicando la simplicidad junto con la autoría colectiva del código y la programación por parejas se asegura que cuanto más grande se haga el proyecto, todo el equipo conocerá más y mejor el sistema completo.

**Fase de exploración**

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

**Fase de planificación**

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

**Fase de iteraciones**

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.

Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores. Wake en [18] proporciona algunas guías útiles para realizar la planificación de la entrega y de cada iteración.

**Fase de puesta en producción**

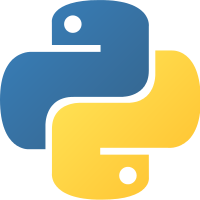
La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

**Lenguaje de programación:**

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

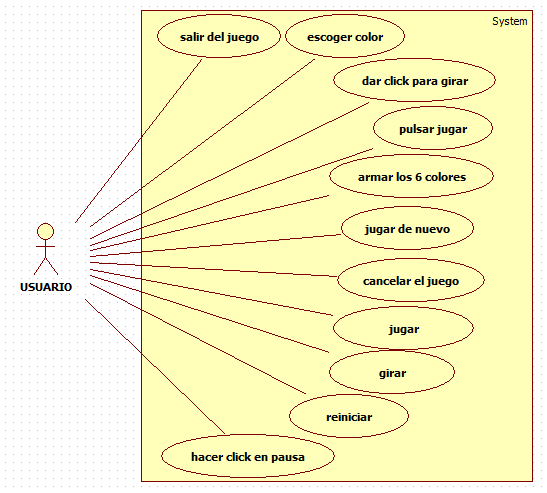
Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License,1 que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.



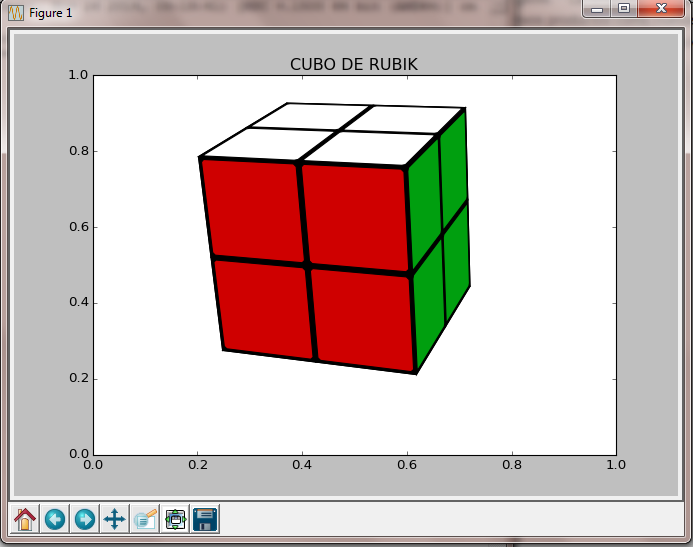
**Diagramas de casos de uso**

La figura 1° nuestra los digrama de casos de uso



**Resultados**

Se observa en la imagen el cubo de rubik 2x2 donde el usuario puede interactuar con el sistema.



**Justificación**

Se propone desarrollar el juego cubo de rubik con el lenguaje de programación Python en que el usuario pueda interactuar y tener la posibilidad de armar el cubo.

Mediante el cubo de rubik podemos desarrollar destrezas memoria, aplicación de algoritmos de resolución y motricidad fina.

**Conclusiones**

Las soluciones del cubo de Rubik pasan por multitud de movimientos estudiosamente coordinados. Estas secuencias tienen que realizarse perfectamente sin ninguna equivocación en los movimientos es asi que la aplicación llega a sistematizar y ayuda a un mejor entendimiento al usuario también se da mayor facilidad de uso.

**Agradecimiento**

Agradecer a nuestra familia por el apoyo que nos brindan día a día ya que sin ellos no pudiéramos desarrollarnos profesionalmente, agradecer también a dios por todo lo bueno que nos brinda.

**Referencias**

**[1]Jose Garrido. (2011). Tutorial método CLL** <http://www.metodogenius.com/academia/descargas/2x2x2/2x2x2%20-%20CLL/TutorialCLL.pdf>

**[2] Stackoverflow.com**

<http://stackoverflow.com/questions/11140163/python-matplotlib-plotting-a-3d-cube-a-sphere-and-a-vector>

**[3]jakevdp,MagicCube**

<https://github.com/jakevdp/MagicCube/blob/master/code/cube.py>

**[4]Progracmacion Extrema xp**

<http://www.ecured.cu/index.php/Extreme_Programming>