

**UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES**

**ESTRUCTURA DE DATOS**

**PROYECTO FINAL**

**“Métodos de ordenamiento, Bejeweled”**

**Nombre de los alumnos:**

Carlos Daniel Cañedo García

Gilberto Román Rojas Durán

**Grupo:** 3° “B"

**Carrera:** Ingeniería en Sistemas Computacionales

**Profesor:** M. en I.A. Eduardo Serna Pérez

**Aguascalientes, Ags. A 16 de diciembre de 2018**

Índice

[**I.** **Resumen Descriptivo** 2](#_Toc532770409)

[**a.** **Fortalezas** 2](#_Toc532770410)

[**b.** **Debilidades** 3](#_Toc532770411)

[**II.** **Temas investigados para la realización del proyecto** 3](#_Toc532770412)

[**a.** **Librería gráfica SDL** 3](#_Toc532770413)

[**b.** **Otros** 3](#_Toc532770414)

[**III.** **Bitácora de trabajo** 5](#_Toc532770415)

[**IV.** **Conclusiones** 6](#_Toc532770416)

[**V.** **Referencias** 6](#_Toc532770417)

# **Resumen Descriptivo**

Bejeweled es un videojuego desarrollado y publicado por PopCap Games. Bejeweled cuenta con varios modos de juego secretos que se pueden desbloquear superando los niveles de la versión Clásico, Puzzle, Acción e Infinito o Sin Fin. El juego fue nombrado en 2001 el Juego Puzzle del año por Computer Gaming World. El objetivo del juego es intercambiar una joya con una adyacente para formar una cadena de tres o más joyas del mismo color. Cuando una cadena se crea, las joyas desaparecen, provocando que aleatoriamente caigan joyas desde la parte superior que tendrá para que las joyas se vayan emparejando. A veces, algunas joyas se combinan automáticamente al caer, a lo que se le denomina cascada.

El proyecto que se presentará en este documento es el proceso de desarrollo de nuestra propia versión, mismo del cual se mencionarán las fortalezas, debilidades y proceso de trabajo que se implementó, así como una bitácora y conclusiones del proyecto y curso en general.

# **Fortalezas**

Se trata de un proyecto que está diseñado para ser lo más óptimo posible, al menos en la manera funcional y gráfica, sentimos que el modelo se *Programación Orientada a Objetos* fue fundamental en lograr este objetivo, debido a que nos permite desarrollar rápidamente sin tener que sacrificar rendimiento, obviamos muchos detalles como los parámetros en las funciones porque **ya no son indispensables**, más si necesarios de vez en cuando; aligera mucho la carga porque facilita la manera en la que leemos el código. Adicionalmente hace que las llamadas a las funciones sean más elegantes.

La estructuración del código es la ideal, cada clase cumple su función, cada función cumple su propósito, no más, no menos, siendo así, nosotros pudimos identificar rápidamente en qué estaba fallando cada sección de código, permitiéndonos solucionarlo de la manera más sencilla y menos dañina posible, tenemos bajo acoplamiento entre clases, el hecho de que sea modular nos permite manejar individualmente cada sección de código y optimizarla sin tener que vernos forzados a modificar aluna otra parte de nuestro código.

Es lo suficientemente divertido, justo como el original, tratamos de hacerlo pegado al sistema original del juego, si existe alguna combinación especial que salga una gema que te permita hacer puntos extra, si bien en calidad gráfica no fue posible obtener una calidad como la del juego original, nos sentimos bastante satisfechos con el resultado obtenido.

¿Quieres leer el código?, ¡no hay problema!, está hecho de tal manera que cualquiera que lo lea lo pueda comprender, claro, sin ir tan lejos, pero los comentarios y el código van de la mano de tal modo que pueda ser comprendido y asimilado sin ningún tipo de problema.

# **Debilidades**

El tiempo de desarrollo fue el óptimo para alcanzar las metas del proyecto, pero no tiene mucho valor agregado, sí, funciona como el original, pero no pudimos ponerle un *plus* que dé la sensación de que tiene nuestro sello personal, algo más para decir: Yo le agregué esta nueva característica.

El código es cansado de leer, si bien es óptimo y se comprende cada función, el nivel de profundidad de llamadas a funciones puede alcanzar un nivel en el que se haga un poco complejo el *traceback*, esto es debido a que muchas clases y métodos funcionan conjuntamente para que el juego funcione, además de usar métodos como composición de objetos.

La librería gráfica es obsoleta, usamos SDL, pero es la versión 1.2, la instalamos para después darnos cuenta de que la 2.0 ya existía, misma que agregaba más y mejores características que la que usamos actualmente, era muy seguro que podíamos mejorar nuestro proyecto con la implementación de esa actualización. Pero implicaba re escribir gran parte del código ya hecho.

# **Temas investigados para la realización del proyecto**

# **Librería gráfica SDL**

El 90% del tiempo de investigación se lo llevó la librería gráfica, afortunadamente, las herramientas y habilidades adquiridas durante los 3 cursos de programación fueron suficientes para generar la lógica que nos permitiera la ejecución del juego.

Desde el hecho de tener que generar una nueva ventana hasta la reproducción de sonido, los temas fueron muy diversos a la par de complejos, tenernos que relacionar con temas como paletas de colores y *offsets* era cosa de locos, pero al final terminamos dominándolo de tal modo que condensamos gran parte del código para gráficos en un header para poderlo implementar sencillamente en próximos proyectos.

Agradecemos mucho a Mr. Foo’, creador del blog ***Lazy Foo*** en el cual vienen tutoriales todo propósito para el uso de la librería SDL, muchas cosas se comprendieron gracias a esos tutoriales, y realmente no pudiéramos haber completado el proyecto de no ser por ese recurso que encontramos por internet.

# **Otros**

El otro 10% fue empleado en temas muy pequeños, tales como maneras sencillas de manejo de binarios o situaciones en las que el código no compilaba y no comprendíamos el error o errores de ejecución que llegábamos a tener.

*Todos los temas investigados serán encontrados en la bibliografía.*

# **Bitácora de trabajo**



# **Conclusiones**

Es otro semestre que termina, y un proyecto siempre es una excelente manera de determinar qué tanto avance hemos tenido en comparación con los semestres anteriores, es una excelente *vara de medir* para ver si realmente estamos avanzando como Ingenieros en Sistemas, y nos sentimos muy contentos con este proyecto. Primero por el cambio de paradigma, los objetos son elementos muy abstractos, pero que potencializan muchísimo la programación funcional.

¿Qué aprendimos?, aprendimos una librería gráfica más poderosa en comparación a la que usamos el semestre pasado, buscábamos algo que fuera más flexible para trabajar y que admitiera sencillamente más formatos multimedia para trabajar, y SDL cumplió esas expectativas.

Aprendimos a hacer animaciones básicas, tales como movimientos de escenarios y de gemas, le dan un aspecto más dinámico al juego, le da una mejor vista.

De manera general, adquirimos habilidad para un mejor manejo de matrices, que es una estructura de datos, el hecho de mapear la matriz una y otra vez nos daba una mejor abstracción para determinar los movimientos necesarios para destruir cada gema y volver a llenar la matriz con nuevas.

Del curso en general queremos decir que fue un curso bastante interesante, existen muchas implementaciones que podemos hacer posibles en C++ con el objetivo de optimizar el rendimiento y memoria de un programa, listas, colas, pilas, grafos, son muchas palabras que se suman al lenguaje técnico que tenemos, combinándolo con POO creemos que son un muy buen complemento que deben tomarse en cuenta para realizar un proyecto de tal magnitud.

(Foo, s.f.) (GameDevGeek, s.f.) (NomNomNom069, s.f.) (SDL\_Dev, s.f.) (Nicholls, s.f.)7

# **Referencias**

Foo, M. (s.f.). *Lazy Foo*. Obtenido de http://lazyfoo.net/SDL\_tutorials/

GameDevGeek. (s.f.). *Animating Sprites SDL*. Obtenido de http://gamedevgeek.com/tutorials/animating-sprites-with-sdl/

Nicholls, P. (s.f.). *Convert SDL\_Key to String*. Obtenido de https://www.pascalgamedevelopment.com/showthread.php?11064-Convert-SDL-key-codes-(SDLK\_ESCAPE-etc-)-to-a-string-(Delphi-2010)

NomNomNom069. (s.f.). *Animating SDL Sprites*. Obtenido de https://www.youtube.com/watch?v=83ReLRqIV7o

SDL\_Dev. (s.f.). *MouseEvents*. Obtenido de https://wiki.libsdl.org/SDL\_MouseButtonEvent