UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

APLICACIÓN MULTIPLATAFORMA PARA LA SIMULACIÓN DE CIRCUITOS DIGITALES

T E S I S

QUE PARA OBTENER EL GRADO DE:
INGENIERO EN COMPUTACIÓN

P R E S E N T A:
PEDRO PABLO BECERRIL CALDERÓN

TUTOR:
ING. VICENTE FLORES OLVERA

MÉXICO, D.F. ABRIL 2014

INDICE

Introducción

Desde la creación de las primeras computadoras se inició el desarrollo de programas automatizados que permiten al usuario la elaboración de tareas complejas de manera eficiente y en un tiempo menor al que tomaría si se hiciera manualmente.

Los primeros programas realizados por el hombre consistían en tarjetas perforadas con secuencias binarias que formaban instrucciones en código máquina. Estas tarjetas se introducían a la computadora y dependiendo de la secuencia en la tarjeta ésta arrojaba un resultado.

Con la evolución de las computadoras a dispositivos electrónicos más sofisticados, los programas pasaron de ser secuencias en tarjetas perforadas a ser archivos electrónicos codificados mediante instrucciones de mayor nivel y con una estructura más comprensible para los seres humanos. En conjunto estas instrucciones se conocían inicialmente como **lenguaje ensamblador**, y fueron la base de los lenguajes de programación de alto nivel que conocemos hoy en día.

La miniaturización de los componentes electrónicos y la evolución de los lenguajes de alto nivel permitieron dar el siguiente gran salto en la evolución de las computadoras y de los programas de las mismas. Los componentes electrónicos de menor tamaño dieron pie a la creación de dispositivos electrónicos inteligentes más pequeños y con capacidades considerablemente altas, por ejemplo los smart phones, tablets y laptops que encontramos actualmente. Estrechamente ligado a lo anterior, los lenguajes de alto nivel dieron pie a la creación de Frameworks y Engines de desarrollo que permitieron acercar el mundo de la programación a un mayor número de personas gracias a su estructura simple y de fácil comprensión.

El objetivo de esta tesis consiste en la creación de una herramienta de simulación portable y de bajo costo de desarrollo que aproveche tanto las capacidades gráficas de los dispositivos móviles inteligentes como las propiedades de despliegue multiplataforma ofrecidas por los engines y frameworks de desarrollo de la actualidad, que esté disponible mínimamente para una plataforma de escritorio (Windows) y una móvil (Android), que facilite además a alumnos, profesores y en general a cualquier persona con conocimientos de sistemas digitales y áreas afines, el desarrollo, estudio, análisis y ejemplificación de circuitos eléctricos a nivel digital mediante una interfaz de usuario amigable, e intuitiva basada en aplicaciones existentes.

En el primer capítulo se presenta al lector un panorama general de la evolución de las computadoras con el fin de brindar un contraste generacional que permita visualizar las capacidades de los dispositivos actuales.

El capítulo dos presenta programas de simulación en versiones de escritorio y móviles

haciendo énfasis en su diseño a fin de ilustrar los elementos gráficos utilizados en programas de éste tipo.

Los capítulos 3 y 4 explican respectivamente el por qué se seleccionó el área de los circuitos digitales y las ventajas que ofrece el software libre en el desarrollo de una aplicación multiplataforma.

El capítulo 5 describe las ventajas del uso de recursos de terceras personas en el desarrollo de una aplicación y el licenciamiento que éste tipo de recursos suele conllevar.

A partir del sexto capítulo se describen temas que están más directamente ligados a la aplicación desarrollada en ésta tesis. En el capítulo 6 se discierne el uso de elementos 2D o 3D como elementos gráficos de la aplicación y el framework o engine como herramienta de desarrollo a seleccionada.

En el capítulo 7 se definen las plataformas objetivo para las que deberá poder desplegarse la aplicación y el licenciamiento respectivo en caso de requerirlo.

El capítulo 8 describe el diseño de la interfaz de usuario final de la aplicación basándose en las interfaces de simulación de programas existentes

Finalmente los capítulos 9 y 10 describen la estructura del programa en cuanto a la forma de manipulación de los elementos interactivos y los resultados obtenidos respectivamente.

A quién va dirigida esta aplicación

La aplicación desarrollada en ésta tesis está pensada como herramienta de apoyo al aprendizaje y por lo tanto está dirigida a un público específico que se encuentre en formación o que cuente con conocimientos en temas como circuitos digitales, lógica booleana y circuitos eléctricos, por ejemplo profesores y alumnos de carreras de ingeniería y electrónica.

Está pensada como un medio de construcción y simulación en tiempo real que permita abatir tiempos de desarrollo mediante la elaboración de un circuito virtual que precede al circuito físico y que además permita el abatimiento costos por desperdicio de material durante el proceso de aprendizaje y ejemplificación.

Al ser multiplataforma la aplicación está orientada a tener un muy amplio alcance, esto con el fin de brindar a todo usuario la misma experiencia de uso independientemente del sistema operativo, pero además garantizando que el aprendizaje no se verá limitado por el tipo de dispositivo al que el alumno o profesor tengan acceso, destacando además el uso de la aplicación en cualquier lugar que el dispositivo lo permita.

Capítulo 1: Dispositivos

Las aplicaciones que encontramos en los distintos marketplaces y en los dispositivos inteligentes de la actualidad están elaboradas con diversas herramientas que van desde lenguajes de alto nivel nativos de los sistemas operativos hasta, Frameworks y Engines multi plataforma. Pero para llegar a estas herramientas tuvo que haber una evolución previa. En este capítulo se describe el proceso evolutivo de los dispositivos móviles con un enfoque más apegado al tipo de software o su equivalente correspondiente a cada etapa de dicha evolución.

1.1 Antecedentes

Las aplicaciones móviles surgen para satisfacer la necesidad de realizar tareas cotidianas de manera eficiente mediante el uso y aprovechamiento de las características de los diferentes dispositivos inteligentes. Este principio se ha venido aplicando desde la invención de las primeras máquinas calculadoras que solventaron la necesidad de realizar cálculos que implican la inversión de mucho tiempo y esfuerzo al hacerse manualmente.

Debido a lo anterior es muy común pensar que el concepto de cómputo está únicamente relacionado al uso de una computadora, cuando en realidad el cómputo viene desde mucho antes de la invención de las mismas.

Cómputo proviene de *computar*, palabra que se origina del latín *computare* que significa contar. Con base en esta definición podemos notar de forma mucho más clara que la necesidad de computar la información surge de más allá de la primera máquina computadora, viene desde la primera vez que el hombre tuvo la necesidad de medir las cosas que le rodeaban, como objetos, distancias e inclusive el tiempo.

Podría decirse entonces que contar fue la necesidad fundamental que dio origen al desarrollo de herramientas que facilitaran dicha labor cuando los dedos de las manos fueron insuficientes. Tales herramientas irían desde el ábaco en un principio, hasta herramientas más complejas con funcionalidad extra como lo son las aplicaciones móviles que conocemos hoy en día.

1.2 Evolución

La evolución de las aplicaciones está estrechamente ligada a la evolución de las computadoras debido a que la forma de desarrollar programas para cada máquina dependía de las funciones y capacidades de la misma. Por este motivo no se puede hablar de la evolución de aplicaciones sin hablar de la de las computadoras.

Desde un principio, la evolución de las computadoras se ha dado en función de la tecnología implementada en los componentes que la integran. Por esto mismo su evolución puede ser dividida en generaciones, las cuales además de ilustrar los

eslabones que han tenido que pasar para llegar a las computadoras actuales, nos permiten apreciar el panorama evolutivo de los componentes electrónicos que hasta hoy en día se utilizan para crear hardware y herramientas con fines cada vez más sofisticados. Ejemplos de estos componentes son los circuitos integrados que se utilizan en la universidades para materias como Sistemas Digitales, materias para la cuales está pensada la aplicación desarrollada en esta tesis.

A continuación se le presenta al lector las diferentes generaciones que han marcado la evolución de la computadora. En cada una de ellas se hace énfasis en el tipo de software disponible en ese momento con el fin de hacer más ilustrativa la evolución que sufrieron los programas computacionales hasta llegar a las aplicaciones móviles que conocemos actualmente.

Primera generación

La primera generación abarca la década de 1950, aproximadamente de los años 1946 a 1956.

Durante esta generación las computadoras se caracterizan por tener un gran tamaño, consumir una considerable cantidad de electricidad y generar una gran cantidad de calor. El medio por el cual se realiza el procesamiento de la información son tubos de vacío, mientras que el esquema de memoria es mediante tambores magnéticos los cuales podían alcanzar hasta 2 kilobytes de capacidad

En ésta generación el equivalente al software son las tarjetas perforadas con secuencias binarias para representar programas. Éstas contenían instrucciones internas y datos para el uso de la máquina. La única forma de programar máquinas de esta generación es mediante código binario.

Segunda generación

La segunda generación se da aproximadamente de los años 1958 a 1964. Durante ésta generación los transistores sustituyeron a los tubos de vacío como medios de almacenamiento y procesamiento de información debido a que los transistores contaban con diversas ventajas entre las cuales destacan un menor consumo de electricidad, menor generación de calor y mayor fiabilidad en comparación con los tubos de vacío. Además fue posible la reducción del tamaño de la máquina.

La forma de almacenamiento durante esta generación consistía en pequeños anillos magnéticos capaces de polarizarse en dos sentidos con el fin de representar un bit.

Los programas de algunas de estas computadoras se escribían en tarjetas perforadas mejoradas, de forma similar a la primera generación, pero además se contaba con la posibilidad de programarlas por medio de un tablero cableado que polarizaba los anillos magnéticos.

El software comenzó a despegar durante esta generación pues se desarrollaron lenguajes de programación como COBOL y FORTRAN.

Tercera generación

La tercera generación de computadoras va de los años 1964 a 1970 y se caracteriza por el uso de circuitos integrados. Estos circuitos permitieron, igual que en la segunda generación, reducir el consumo de electricidad, la emisión de calor y el tamaño en general de la máquina.

El almacenamiento y procesamiento de la información se realiza por chips. Los chips son piezas de silicio contenedoras de componentes electrónicos miniatura, dichos componentes son capaces de almacenar información dentro de sí mismos en forma de cargas eléctricas.

La forma de programar aplicaciones para estas máquinas es mediante lenguajes de programación de bajo nivel. Es durante esta generación que se populariza la industria del software. Los medios físicos de programación como las tarjetas perforadas dejan de utilizarse gracias a la propiedad de los circuitos integrados de almacenar información en forma de carga.

Cuarta generación

La cuarta generación abarca los años 1971 a 1988. Los adelantos en microelectrónica continúan su curso y surgen los microprocesadores, que son circuitos integrados de alta densidad capaces de reunir los elementos básicos de una máquina dentro de sí y que permiten el surgimiento de las computadoras personales.

El almacenamiento y procesamiento de la información se realiza de igual manera mediante chips. Además surgen chips que pueden cumplir con distintas finalidades.

La programación durante esta generación se hace mediante lenguajes de nivel medio, los cuales presentan palabras clave un poco más específicas pero que aún requieren de cierta preparación técnica para su implementación.

El término software se vuelve común para referirse a los programas computacionales.

Quinta generación

La quinta generación de computadoras va de los años 1984 a 1999 aproximadamente. Durante este periodo el ordenador personal ya es una realidad que se vive cotidianamente gracias a que el desarrollo de la microelectrónica sigue haciéndose presente.

La industria de software se siente rezagada por el desarrollo tecnológico e intenta ponerse la par de tales avances mediante el impulso al desarrollo de programas y sistemas con los que se manejan las computadoras. Se fomenta así el desarrollo de lenguajes que tengan como principales características palabras clave más cotidianas en vez de códigos específicos y como resultado aparecen lenguajes de más alto nivel que cuentan con una sintaxis más sencilla y más comprensible no solamente para el programador sino también para cualquier persona sin una preparación técnica.

Sexta generación

La sexta generación de computadoras se considera la que vivimos actualmente y que está en marcha desde el año 1999. Las características principales que describen esta generación de computadoras son entre otras cosas la integración de microprocesadores capaces de realizar el cálculo de miles de millones de operaciones por segundo, interfaces de usuario más complejas visualmente pero más sencillas de manipular, además de la integración de hardware y dispositivos externos que le dan mayor funcionalidad a la máquina o computadora.

La programación en esta generación ha evolucionado al grado del uso de lenguajes de muy alto nivel los cuales cuentan con una estructura más simple y comprensible, que utiliza palabras clave más parecidas al lenguaje natural.

Se popularizan y se ponen a disposición del público los Software Development Kits (SDKs) de los sistemas operativos móviles de mayor influencia en el mercado. Su estructura le permite acercar el mundo del desarrollo de aplicaciones a prácticamente cualquier persona con conocimientos básicos de programación.

También surgen los engines de desarrollo que van más allá de solamente proporcionar las librerías para programar para diferentes sistemas operativos con un código genérico, proveen además una interfaz de usuario con elementos gráficos que facilitan increíblemente el diseño de la aplicación en cuestión.

Generación	Software o equivalente
Primera	Tarjetas perforadas con secuencias binarias
Segunda	Tarjetas perforadas mejoradas
Tercera	Lenguajes de bajo nivel
Cuarta	Lenguajes de nivel medio
Quinta	Lenguajes de alto nivel
Sexta	Lenguajes de muy alto nivel

Tabla 1: Generaciones de computadoras y su equivalente en software.

Capítulo 2: Software de simulación existente

El software de simulación surge a partir de que las computadoras adquirieran la capacidad de computar elementos gráficos, esto es durante la quinta generación de computadoras, la cual comienza en 1984.
En el caso específico de software de simulación de circuitos destacan por su gran aceptación y uso 3 programas, dos en versiones para escritorio y una aplicación móvil. Estos son: Proteus, Xilinx y Every Circuit respectivamente. La aplicación desarrollada en esta tesis está basada en el análisis de las diferentes características de estos tres simuladores.
A continuación se presenta al lector una descripción a grandes rasgos de cada uno de los programas de simulación anteriormente mencionados.

2.1 Versión de escritorio

Proteus y Xilinx son dos programas de simulación de escritorio pertenecientes a *Labcenter Electronics* y *All programmable technologies from Xilinx* respectivamente. Ambos programas ofrecen una interfaz gráfica con la cual se pueden crear y simular circuitos en tiempo real. Hasta ahora no han surgido versiones móviles de ninguno de los dos.
Se seleccionaron estos dos programas como base para el diseño de la aplicación desarrollada en esta tesis debido a que durante la carrera de ingeniería, en materias relacionadas con circuitos digitales, se utilizan constantemente éstos programas, además cuentan con las características más destacables y útiles que un simulador de circuitos debe contener.

A continuación se da una breve descripción de cada programa con el fin de ilustrar las características de cada uno.

2.1.1 Proteus

Proteus es un software compuesto por dos sub programas principales llamados ARES (Advanced Routing and Editing Software) e ISIS (Intelligent Schematic Input System). ARES es la parte encargada de las herramientas utilizadas para la elaboración de circuitos impresos, mientras que ISIS se encarga del diseño del plano representativo del circuito eléctrico mediante el uso de componentes variados que van desde resistencias hasta microcontroladores.

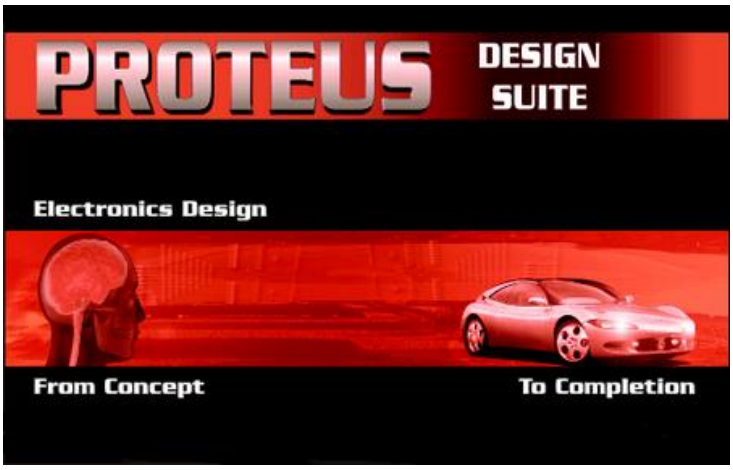


Figura 1: **Portada del programa Proteus de Labcenter**

ISIS cuenta con una extensa librería de componentes virtuales que van desde los más básicos como transistores y compuertas lógicas hasta los más avanzados como microcontroladores. Los componentes más complejos de ISIS tienen la capacidad de configurarse virtualmente de manera muy similar a como se configurarían en la vida real, por ejemplo, a los microcontroladores se les puede cargar virtualmente el archivo de código fuente que determinará su comportamiento, mismo código que se utiliza para programar a su equivalente microcontrolador real.

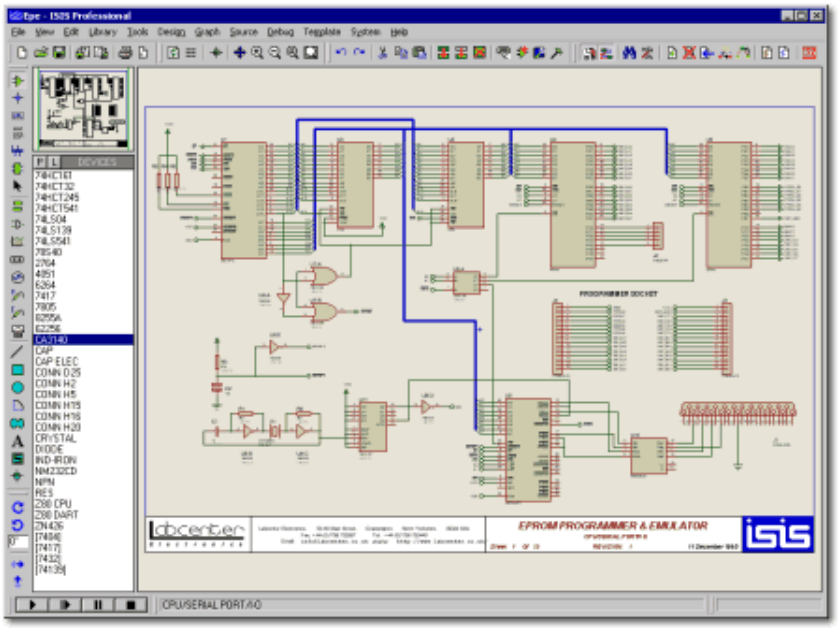


Figura 2: **Ejemplo de un circuito elaborado con el componente ISIS de Proteus**

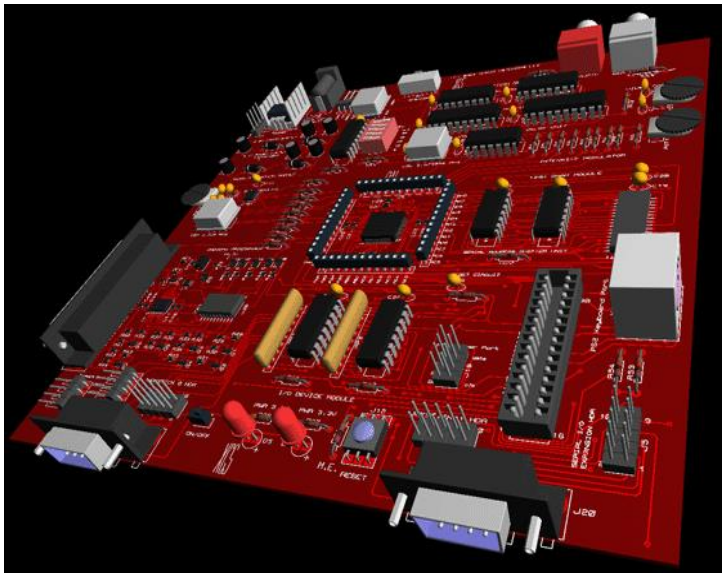


Figura 3: Ejemplo de un circuito impreso elaborado con el componente ARES de Proteus

2.1.2 Xilinx

Xilinx es una empresa que se dedica al desarrollo de circuitos integrados conocidos como FPGAs (siglas en inglés de Field Programmable Gate Arrays). Los FPGAs son dispositivos que contienen bloques de lógica programable configurables mediante lenguajes especializados. Estos dispositivos dependiendo de su programación pueden llegar a representar funcionalidades que van desde una compuerta lógica hasta sistemas más complejos y todo en el mismo chip.

El software Xilinx es más parecido a un entorno de programación que a un simulador gráfico de circuitos, pero su interfaz también cuenta con la capacidad de representar resultados en forma de gráficas.

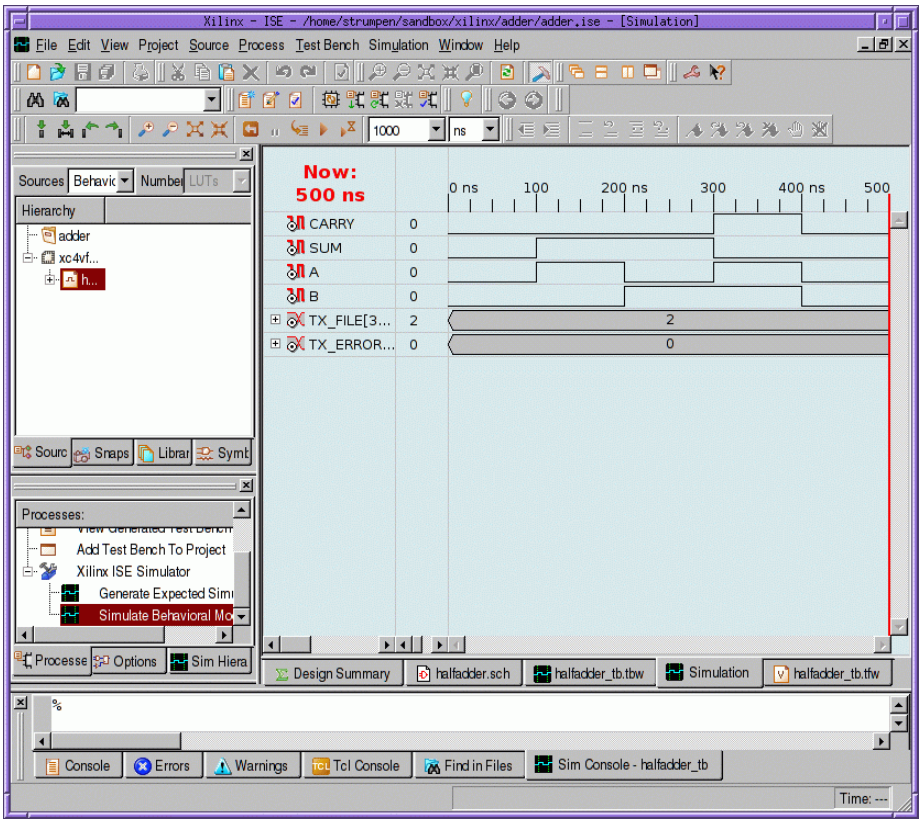


Figura 4: **Interfaz de Xilinx representando variables.**

2.2 Every Circuit (Móvil)

Every Circuit es una aplicación para dispositivos móviles con el sistema operativo Android. Cuenta con una interfaz gráfica similar a la del componente ISIS de Proteus pero con elementos más grandes y simples que aprovechan la propiedad táctil de los dispositivos móviles para su manipulación.

Los componentes electrónicos disponibles en la aplicación son más limitados que los de Proteus y están más orientados al desarrollo de circuitos eléctricos en general, no solamente circuitos digitales.

La interfaz de Every Circuit también brinda la posibilidad de mostrar curvas gráficas representativas de las distintas variables existentes en un circuito, por ejemplo corrientes y voltajes.

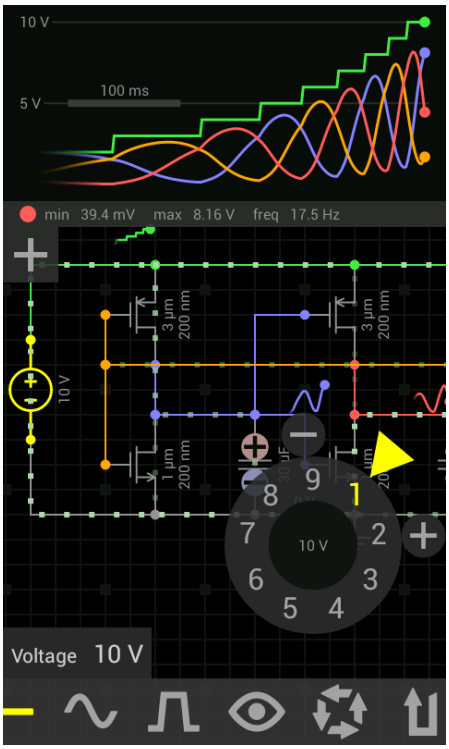


Figura 5: Vista de la interfaz gráfica de Every Circuit

Debido a la falta de referencias no es posible decir a ciencia cierta si la aplicación fue creada por una sola persona, por un grupo de desarrolladores o por una compañía. Every Circuit está registrada en Google Play bajo el Developer ID de *MuseMaze* y la única referencia disponible es la siguiente dirección:

http://www.everycircuit.com/index.html

En dicha dirección hay un pie de página que se refiere a MuseMaze como MuseMaze Inc. Pero tampoco se puede asegurar que sea una compañía debido a que la página describe exclusivamente a la aplicación Every Circuit y no parece haber otros desarrollos bajo éste ID en Google Play.

Como conclusión de este capítulo, las aplicaciones aquí descritas son útiles porque siguen mejorando desde que fueron creadas, sin embargo, con excepción de Every Circuit, todas fueron creadas antes de que el desarrollo móvil tuviera la aceptación y facilidad con la que se cuenta actualmente. Esto no implica una desventaja pero define la posibilidad del desarrollo de una aplicación con características similares que aproveche todas las ventajas del desarrollo móvil y multi plataforma que se tienen actualmente.

Capítulo 3: Por qué circuitos Digitales

Los circuitos eléctricos se dividen en dos áreas principales: de los circuitos analógicos y de los circuitos digitales. El área analógica se encarga del estudio de magnitudes continuas como los voltajes y corrientes, mientras que el área digital estudia el comportamiento de los circuitos basados en valores discretizados representados por 1s y 0s.

Se decidió desarrollar una aplicación específicamente para el área digital debido a que se tiene como objetivo apoyar a materias relacionadas con el diseño de sistemas digitales, aprovechar el despliegue multiplataforma ofrecido por las herramientas de desarrollo de software actuales e innovar para no desarrollar algo ya existente.

Existen muchas herramientas de simulación de circuitos, sin embargo la mayoría de estas se centran en el área analógica y carecen de soporte multiplataforma; la aplicación desarrollada en esta tesis pretende cubrir dichos aspectos y además aprovechar las capacidades de procesamiento de los dispositivos inteligentes actuales para obtener un alcance mayor que beneficie a los usuarios finales.

3.1 Analógico vs Digital

A pesar de que la aplicación está orientada a los circuitos digitales hay conceptos básicos que se deben conocer para poder comprender la forma en que se relacionan estas dos áreas. A continuación y a lo largo de éste capítulo se describen los sistemas analógicos y digitales y algunos otros conceptos con el fin de proporcionar al lector los términos básicos que se utilizarán a menudo a lo largo de ésta tesis.

3.1.1 Sistema Analógico

Un sistema analógico es aquél que contiene magnitudes con valores continuos. Tomando como ejemplo una magnitud de la vida cotidiana podemos considerar a la distancia una magnitud analógica. Supongamos que medimos la distancia que caminamos a lo largo de la mañana y definimos un intervalo observación de 6am a 7am. Durante éste intervalo la distancia recorrida no varía instantáneamente de las 6am a las 7am sino que hay una infinidad de valores a diferentes instantes.

La gráfica de valores continuos representativa de la distancia recorrida durante la mañana sería como la siguiente:

Figura 6: Gráfica de un sistema analógico.

3.1.2 Sistema Digital

Un sistema digital es aquél que contiene magnitudes con valores discretizados, es decir, valores que se contabilizan cada cierto intervalo. Si retomamos el ejemplo de la distancia recorrida durante la mañana podemos discretizar los valores para tomar solamente la distancia a cada hora, de esta manera podemos convertir una magnitud analógica a su equivalente representado por un código digital.

La gráfica discretizada de la distancia recorrida quedaría representada de la siguiente manera con cada punto representado por un código digital que consta de 1s y 0s:

Figura 7: Gráfica de un sistema digital

3.2 Análisis a nivel digital

La electrónica digital hace uso de dos estados, ENCENDIDO y APAGADO, para representar sus valores. Éstos son la interpretación de dos niveles de voltaje diferentes: ALTO (HIGH) y BAJO (LOW).

Debido a que son solamente dos, la forma de representarlos llega a ser muy variada y puede realizarse por ejemplo mediante la medición de niveles de corriente o por los relieves en un DVD.

A las combinaciones de varios de estos estados se les conoce como **códigos**. Los códigos se utilizan para representar diversos tipos de datos como por ejemplo caracteres alfanuméricos y otros símbolos.

Al sistema de numeración que se rige por únicamente dos estados se le conoce como sistema de numeración **binario** y emplea los dígitos 1 y 0 para su representación. A cada uno de estos dígitos por separado se les conoce como **bits**.

3.2.1 Niveles lógicos

Los niveles lógicos son los niveles de voltaje que se utilizan para representar un 1 o un 0 en un circuito digital. Pensando idealmente, un voltaje V1 puede representar un nivel ALTO y otro voltaje V2 un nivel BAJO, sin embargo en un circuito real no hay forma de ser precisos en un determinado voltaje, es por esto que para la representación de un nivel ALTO o BAJO se toman los valores de voltaje de un rango comprendido entre un mínimo y un máximo. Este rango definido no debe solaparse para cada nivel a fin de evitar confusión entre niveles lógicos, aquellos valores comprendidos fuera de cualquiera de estos dos rangos no serán aceptados para garantizar un funcionamiento correcto.

La siguiente figura representa la forma en que deben definirse los rangos para cada nivel lógico.

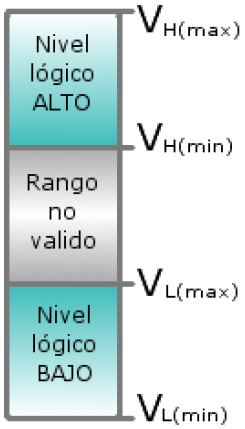


Figura 8: Rangos de niveles lógicos

Los rangos aceptables están representados por las áreas definidas por VH(max) y VH(min) para el nivel lógico ALTO y por VL(max) y VL(min) para el nivel BAJO.
En circuitos digitales reales el rango ALTO va de 2V a 3.3V y el rango BAJO de 0V a 0.8V.

3.2.2 Dígitos binarios

Como se mencionó anteriormente a los dígitos del sistema **binario** se les denomina **bits**. Los bits son representados por los dígitos 1 y 0 y son utilizados en los circuitos digitales para representar niveles de voltaje altos o bajos dependiendo convenio adoptado. Existen dos convenios a los que se les conoce como **lógica positiva** y **lógica negativa**. En el convenio de **lógica positiva** se utilizan 1s para representar un nivel lógico ALTO y 0s para un nivel BAJO. Mientras que en el convenio de **lógica negativa** los 1s representan un nivel BAJO y los 0s un nivel ALTO.

3.3 Aplicaciones de los sistemas digitales

Los sistemas digitales actualmente son utilizados en infinidad de aplicaciones que van desde juguetes hasta herramientas y maquinaria. En todos los casos el conjunto de varios módulos digitales con diferente funcionalidad es el que le da vida al sistema completo y a la aplicación en sí.
Una aplicación digital por lo tanto se define como un sistema que cuenta con múltiples módulos que cumplen diferentes fines.
A continuación se presenta un sistema de ejemplo de un teclado numérico que muestra los dígitos en un display.

Figura 9: **Ejemplo de sistema digital por bloques**

El sistema se conforma por cuatro bloques: teclado numérico, un codificador, un decodificador y un display. El teclado numérico envía al codificador una serie de bits dependiendo de la tecla presionada, el decodificador traduce el código recibido a su equivalente numérico binario y finalmente el codificador convierte el código del decodificador a otro código diferente interpretable por el display como un caracter numérico.

Capítulo 4: Por qué utilizar software libre

4.1 Definición de software libre

El software conforma la parte lógica de una computadora y brinda la funcionalidad que ésta necesita para realizar diversas tareas que pueden ir desde simple procesamiento de texto hasta cálculo de grandes cantidades de información.

Software es un anglicismo que se utiliza para referirse a los programas computacionales. Estos programas se componen de cientos de líneas de código de programación las cuales determinan el funcionamiento del programa.

Software Libre se refiere a todos aquellos programas cuyo código está disponible para que cualquier usuario pueda hacer uso de él, modificarlo o distribuirlo sin que esto implique problemas legales. A estos programas también se les conoce como programas de código abierto. A su contraparte, es decir el software cuyo código no está disponible abiertamente y por el cual hay que pagar licencia, se le conoce como *Software Comercial* o *Software Privativo*, siendo éste último término utilizado mayoritariamente por personas afines al Software Libre.

Cabe mencionar que en algunas ocasiones el Software Libre llega a confundirse con el *Freeware*. Esta confusión radica en la traducción literal del inglés *Free Software* donde *Free* puede traducirse como libre o como gratuito, sin embargo *freeware* es el término que se utiliza para referirse a programas de distribución sin costo, pero no necesariamente de código abierto.

Se puede reconocer al Software Libre porque éste debe cumplir con 4 libertades principales que caracterizan a todo programa de éste tipo. Estas libertades fueron enunciadas por Richard Stallman fundador del *Proyecto GNU*, proyecto orientado a la creación de un sistema operativo no privativo, y de la *Free Software Foundation* (FSF), fundación no lucrativa que promueve el uso de software libre y que defiende los derechos de los usuarios de éste tipo de programas computacionales:

Libertad 0: Libertad de usar el programa para cualquier propósito

Libertad 1: Libertad de estudiar el código fuente y modificarlo para adaptarlo a las necesidades.

Libertad 2: Libertad de redistribuir copias con el fin de ayudar al prójimo.

Libertad 3: Libertad de distribuir copias de sus versiones modificadas a terceros para el beneficio común.

Figura 10: Logo de la Fundación de Software Libre.

El «Software Libre» es un asunto de libertad, no de precio. Para entender el concepto, debe pensarse en «libre» como en «libertad de expresión», no como en «cerveza gratis».

4.2 Ventajas

Costo: El Software Libre tiene la característica de poder distribuirse a un bajo costo, gratuitamente o a cambio de una donación voluntaria, ésta característica permite reducir enormemente los costos de producción de una aplicación.

Libertad de uso: Es posible utilizar software libre para crear software comercial y generar ganancias, dichas ganancias serían completamente del autor o autores de la aplicación ya que no hay que pagar licencias por el uso de las herramientas o por la liberación de la aplicación.

Flexibilidad: Como se mencionó anteriormente el software de código abierto está disponible para que cualquier persona lo modifique, esto lo hace multidisciplinario y le da la posibilidad de mejorar a un ritmo más acelerado.

Multiplataforma: El software de código abierto se caracteriza por estar disponible para diferentes sistemas operativos. Ésta característica trae consigo la ventaja de que cualquier archivo o recurso producto de un software libre sea también funcional en cualquier plataforma.

Tamaño de la aplicación: Las herramientas hechas con software libre llegan a tener la característica de utilizar las mismas librerías para cumplir diferentes propósitos en aplicaciones similares, en consecuencia la aplicación final se vuelve mucho más ligera comparada con otras comerciales. Esto se debe a la reutilización del código de las librerías compartidas.

4.3 Desventajas

Documentación: La documentación correspondiente al software libre es mucho menos detallada que la del software comercial debido a que no es una obligación de todos los colaboradores el tener buenas prácticas al respecto.

Garantía: No existe garantía por parte del autor de que el software se comporte como debe o que tenga el comportamiento más eficiente.

Instalación: Algunas aplicaciones llegan a requerir cierto conocimiento especializado que puede volver complicada su instalación. Incluso los requerimientos llegan a variar según la plataforma en la que se vaya a utilizar el programa por lo que el proceso de instalación puede cambiar de plataforma a plataforma.

4.4 Tipo de software requerido

La aplicación de ésta tesis consiste en una interfaz gráfica manipulable capaz de construir y simular circuitos digitales en tiempo real. Ésta se encuentra conformada por dos módulos principales encargados de la parte gráfica y la parte lógica. Para la construcción de estos módulos se requiere de software específico con el cual se pueda generar el contenido necesario de la aplicación.

Para la parte gráfica es necesario software de diseño y edición que cuente con la capacidad de generar contenido visual.

En cuanto a la parte lógica, se necesita software que permita codificar la lógica del programa e integrar los elementos visuales con determinado comportamiento.

4.4.1 Software de diseño

A continuación se presenta una lista de los tres programas gratuitos de edición de imágenes con mayor aceptación y una breve descripción de los mismos.

1. GIMP

Programa de edición de imágenes gratuito y de código abierto. Cuenta con una gran cantidad de herramientas y efectos precargados que compiten con programas comerciales profesionales.

Disponible en versiones para Windows, Linux y Mac. Además cuenta con una versión portable capaz de ejecutarse desde una memoria USB.

Figura 11: **Portada de GIMP V2.8**

2. Paint.NET

Editor de fotografías freeware disponible únicamente para el sistema operativo Windows. Se caracteriza por tener una interfaz gráfica sencilla parecida a la de GIMP y por contar con una gran variedad de efectos y herramientas de edición.

Surgió como un proyecto universitario apoyado por Microsoft y pretende ser un sustituto gratuito de *Microsoft Paint*, programa básico de edición que viene integrado con todas las versiones del sistema operativo Windows.

Figura 12: **Logo de Paint.net**

3. Pixlr

Pixlr es una aplicación web gratuita de las más populares disponibles en línea. Permite editar y manipular imágenes digitales con una variedad de herramientas y una interfaz mucho más simplificada que las aplicaciones anteriormente descritas.

Fue creada originalmente por Ola Sevandersson y posteriormente adquirida por Autodesk en el 2011.

Actualmente Pixlr cuenta con una aplicación para smartphones y tablets que le permite ejecutarse en dispositivos móviles.

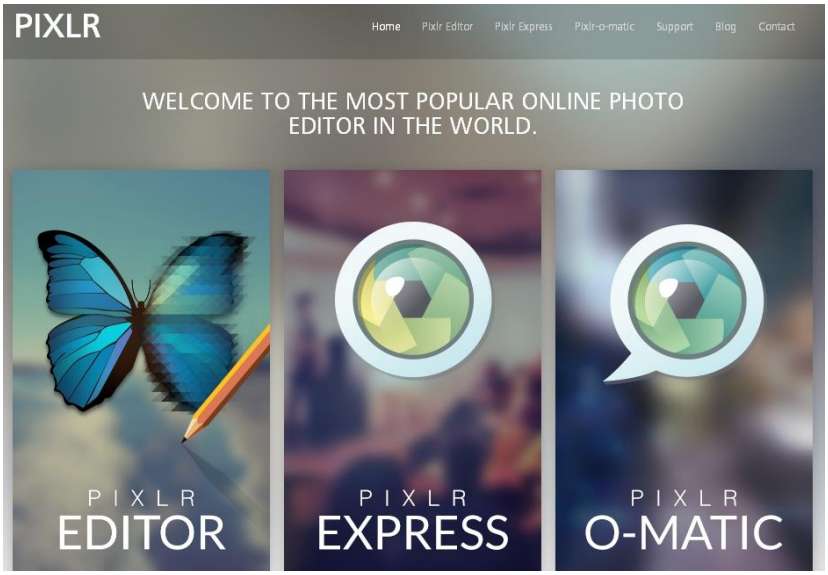


Figura 13: Página web de PIXLR

4.4.2 Software de programación

Al software utilizado para programación, edición, compilación y ejecución de código se le conoce como *Entorno Integrado de Desarrollo* o IDE por sus siglas en inglés de *Integrated Development Environment*.

Un IDE es un programa de computadora que se caracteriza por tener un área tipo editor de texto plano en la que se puede ingresar código en algún lenguaje de programación y una interfaz gráfica con la cual se pueden acceder a las diferentes herramientas que vienen integradas con el programa las cuales que facilitan algunas tareas al programador como por ejemplo la detección de errores de sintaxis y búsqueda de palabras específicas dentro del código.

A continuación se enlistan los IDEs mejor valorados para la programación en diferentes lenguajes.

Netbeans

Netbeans es el IDE de programación de Sun Microsystems. Está orientado principalmente para el desarrollo en el lenguaje Java pero también cuenta con soporte para otros lenguajes como PHP y C/C++ en diferentes versiones del IDE.

Netbeans está escrito en Java lo que lo hace ejecutable en cualquier plataforma que sea capaz de ejecutar una máquina virtual de Java o *JVM* (*Java Virtual Machine*).

Cuenta con licencia GPL, es decir es software libre.

La interfaz gráfica de Netbeans brinda diferentes herramientas que facilitan el proceso de codificación, depuración y prueba de la aplicación, pero además se distingue por tener una interfaz gráfica modular especializada tipo *drag & drop* que simplifica el

desarrollo de aplicaciones más visuales como aquellas en las que predominan interfaces de usuario como ventanas, botones, cajas de texto, etc.

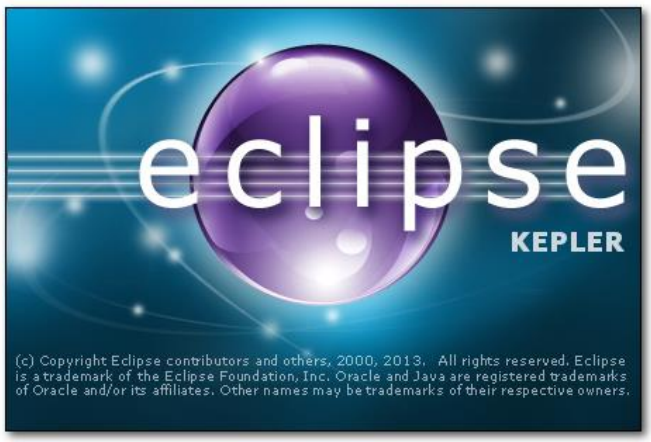
Figura 14: Logo de NetBeans

Eclipse

Eclipse es un IDE de programación muy flexible desarrollado por la *Fundación Eclipse*. Se distingue por ser multiplataforma, de código abierto y por soportar diferentes lenguajes de programación en diferentes versiones del IDE.

Cuenta con herramientas que le facilitan al programador el proceso de codificación y cuenta con la capacidad de instalar complementos que le brindan funcionalidad extra.

La interfaz de Eclipse es muy similar a la de Netbeans lo que permite una más sencilla adaptación para los usuarios de éste otro IDE, además puede modificarse dependiendo de los complementos que se tengan instalados y de la modalidad en la que se trabaje.



Portada de Eclipse IDE

Visual Studio

Visual Studio es un entorno de desarrollo exclusivo para sistemas operativos Windows. Permite crear aplicaciones para diferentes plataformas pero está más orientado a aquellas que son propiedad de Microsoft. Al igual que otros IDEs soporta diferentes lenguajes de programación que van desde los más complejos como C/C++ hasta algunos más básicos como HTML o JavaScript. Además, este IDE cuenta con soporte

para el framework *.NET*, también desarrollado por Microsoft, el cual consta de una amplia librería de funciones en lenguajes interoperables, es decir, lenguajes que pueden usar código escrito en otros lenguajes.

Figura 15: Logo de Visual Studio

4.5 Selección final de software libre

Se decidió utilizar únicamente software libre para aprovechar dos de las ventajas que éste ofrece y que impulsan los objetivos de la aplicación de esta tesis. Dichos objetivos son: el abatimiento de costos de desarrollo y el despliegue en diferentes sistemas operativos.

Software de diseño.

El software seleccionado para el diseño de los elementos visuales de la interfaz de la aplicación es conocido como *GIMP* (GNU Image Manipulation Program), un programa de código abierto de edición de imágenes que pertenece al proyecto GNU.

El programa cuenta herramientas y con una interfaz similares a las del software comercial Photoshop de Adobe y ha tenido un éxito tal que se ha llegado a considerar como la alternativa gratuita a Photoshop.

El contenido generado con esta aplicación es de gran calidad y ha llegado a ser utilizado por artistas profesionales, algunos de los cuales exponen su trabajo en la *Libre Graphics Meeting*, conferencia que reúne proyectos de código abierto del área de gráficos, desarrolladores y usuarios para compartir desarrollos e ideas para mejorar el software y mostrar lo que se puede lograr con éstos.

Figura 16: Logo de GIMP (GNU Image Manipulation Program)

http://www.gimp.org/

Software de programación.

En cuanto al software de programación se seleccionó Eclipse, un IDE multiplataforma con licencia de software libre (Eclipse Public Licence) que ofrece diversas herramientas de programación de código abierto.
El Proyecto Eclipse fue desarrollado originalmente por IBM en el 2001. La *Fundación Eclipse* fue creada en el 2004 como una corporación no lucrativa para que estuviera a cargo del desarrollo de éste software y de la *Comunidad Eclipse* la cual está compuesta por todos aquellos individuos y organizaciones que deseen colaborar en el desarrollo de éste y otro software libre comercialmente amigable.
Cabe mencionar que éste IDE es el más utilizado y mejor documentado al utilizar librerías o frameworks de terceros ya que es muy adaptable. Google por ejemplo distribuye una copia modificada de Eclipse en su *ADT Boundle*, paquete de software para desarrolladores de Android que consta del IDE más el SDK de Android con la última plataforma liberada.

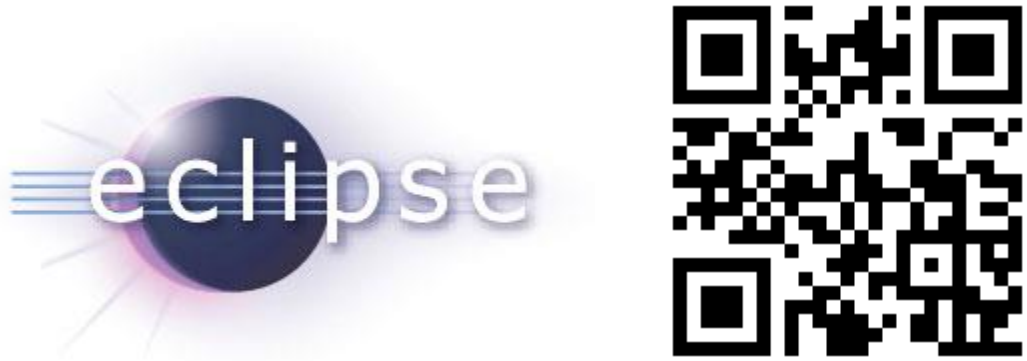


Figura 17: Logo de Eclipse

https://www.eclipse.org/

Capítulo 5: Recursos de terceros

5.1 ¿Qué son?

Los recursos de terceros como el nombre lo indica son todos aquellos recursos cuya autoría no es propia sino de otras personas. Estos recursos pueden ser gratuitos o pueden tener algún costo dependiendo de las características que los conforman, como por ejemplo el nivel de detalle o la calidad del trabajo.

Los recursos de terceros nos ayudan a eficientar el desarrollo de proyectos ya que no hay que generar el contenido nosotros mismos, sin embargo existe una desventaja y es que en ocasiones éstos no cubren completamente las especificaciones deseadas por lo que la alternativa es elaborar los recursos por cuenta propia si se tiene la capacidad o bien contratar a alguien que los elabore con las especificaciones necesarias.

Los recursos de terceros tienen diferentes tipos de licencias que se pueden englobar en comerciales y gratuitas.

Las licencias comerciales requieren del pago de cierta cantidad monetaria específica para poder hacer uso de los recursos deseados ya sea por un tiempo limitado o de por vida.

Las licencias gratuitas tienen términos de uso especificados por el autor de los recursos, éstos son establecidos en términos legales para proteger el trabajo de las personas pero sin exigir ningún tipo de remuneración. La ganancia al aplicar este tipo de licencia se ve reflejada en el renombre que recibe el autor por el uso de su trabajo en diferentes proyectos ya que en la mayoría de estos casos lo único que se exige para poder utilizar libremente los recursos de licencia gratuita es una atribución, es decir una referencia al autor original en algún lugar del proyecto, comúnmente en el área de créditos.

Los recursos de terceros que se utilizarán en la aplicación de esta tesis son aquellos que tienen que ver con la tipografía implementada y los sonidos utilizados en la interfaz gráfica para la retroalimentación del usuario. Se dará preferencia a aquellos recursos que se encuentren bajo licencia Creative Commons debido a que ésta incentiva el compartimiento de la información.

5.2 Licencia Creative Commons

Creative Commons es una organización sin ánimo de lucro que brinda licencias de derechos de autor gratuitas por medio de herramientas legales. Dichas licencias permiten al autor de una obra o recurso compartirla voluntariamente al público bajo los

términos que él o ella elija y sin que esto cause problemas de derechos de autor, contando además con la posibilidad de cambiar estos términos en cualquier momento.

Figura 18: Logo Creative Commons

La licencia Creative Commons, abreviada CC, puede trabajar en conjunto con la licencia *Copyright* para modificar los términos de ésta última de acuerdo a las necesidades que se requieran y de esta manera utilizar el término “algunos derechos reservados” en vez de “todos los derechos reservados”.
La finalidad de CC es la de maximizar la creatividad y fomentar la innovación por medio de la ruptura de las barreras que imponen los términos de copyright sin que esto implique un robo de la propiedad intelectual.

Our mission

Creative Commons develops, supports, and stewards legal and technical infrastructure that maximizes digital creativity, sharing, and innovation.

Our vision

Our vision is nothing less than realizing the full potential of the Internet — universal access to research and education, full participation in culture — to drive a new era of development, growth, and productivity.

Misión y visión de Creative Commons
https://creativecommons.org/about

Ésta licencia, como se puede apreciar en su misión y visión, promueve el uso compartido de diferentes tipos de recursos como lo son los literarios, musicales, gráficos, etc. Pero hace énfasis en los recursos digitales disponibles en Internet. Esto se debe a que el Internet es el elemento que nos permite tener acceso a la información de manera universal y a su vez nos brinda herramientas para compartirla como copiar, pegar o compartir en redes sociales. Sin embargo, damos por hecho que el uso de estas herramientas no tiene consecuencias legales ya que ignoramos si alguno de los recursos de internet tiene limitaciones impuestas por licencias de uso como por ejemplo

las del *Copyright*. Cabe mencionar que el *Copyright* surge mucho antes de que el internet alcance la aceptación que tiene actualmente por lo que mucha de la información puede estar protegida.
Aquí es donde Creative Commons entra en acción para facilitar el acceso a la información, permitiendo modificar los términos bajo los que se rige una obra o recurso que se encuentra registrado por copyright.
La estructura de una licencia Creative Commons se compone de 3 capas:

- Código Legal

Esta capa es el tradicional instrumento legal que contiene los términos de la licencia en un lenguaje mucho más formal utilizado por los abogados para determinar los términos aplicados por esta licencia.

- Legible por humanos

En esta segunda capa está orientada a todas aquellas personas que no están tan familiarizadas con la terminología legal utilizada en la primera capa. Contiene el resumen de la licencia en un lenguaje menos formal. A este resumen también se le conoce como el *Commons Deed*.

- Legible por máquinas

Esta capa contiene un resumen de los derechos clave contenidos en la licencia en un formato estandarizado y comprensible por diferentes tipos de tecnología como lo son los motores de búsqueda. A este formato se le conoce como *CC Rights Expression Language (CC REL)*.

Como se mencionó anteriormente una licencia CC permite al autor de una obra elegir entre diversos términos y condiciones para licenciar su trabajo, los diferentes tipos de licenciamiento que ofrece CC se enlistan a continuación. Una descripción más detallada se puede encontrar en la página de Creative Commons en la dirección: http://creativecommons.org/licenses/

- Atribución (CC BY)

Permite a otros distribuir, mezclar, retocar y crear a partir de la obra licenciada con o sin fines comerciales siempre que se dé crédito a la creación original. Este tipo de licencia es la que provee más libertad de uso de la obra licenciada por lo que se recomienda para todos aquellos recursos cuyo objetivo sea tener la mayor difusión y utilización.

- Atribución - sin derivados (CC BY-ND)

Permite la distribución comercial o no siempre que la obra original circule íntegra y sin alteraciones dando además crédito al autor original.

- Atribución - compartir igual (CC BY-SA)

Permite a otros crear obras a partir de la obra original, incluso con fines comerciales, siempre y cuando se atribuya crédito al autor original y además dichas nuevas creaciones sean licenciadas bajo los mismos términos de licencia que la obra en la que fue basada.

- Atribución - no comercial (CC BY-NC)

Permite la creación y distribución a partir de una obra original de manera no comercial, se debe atribuir al autor original y además las obras derivadas deben permanecer con carácter no comercial pero pueden licenciar dichos derivados bajo diferentes condiciones.

- Atribución - no comercial - compartir igual (CC BY-NC-SA)

Permite crear y distribuir a partir de una obra de manera no comercial, se debe atribuir al autor original y además licenciar los derivados bajo las mismas condiciones de la obra base.

- Atribución - no comercial. sin derivados (CC BY-NC-ND)

Este tipo de licencia permite descargar y compartir una obra atribuyendo siempre al autor original, sin embargo no podrá comercializarse ni modificarse de forma alguna.

Todo trabajo licenciado con alguno de los términos de Creative Commons descritos anteriormente se caracteriza por incluir un gráfico descriptivo del tipo de licencia bajo el que se encuentra registrado como los que se ven a continuación.

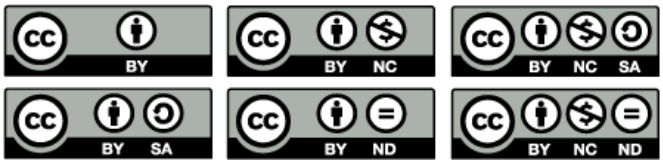


Figura 19: Graficos de licencias Creative Commons

5.3 Páginas con recursos gratuitos

Existen infinidad de páginas disponibles con recursos de terceros, en muchas de ellas se pueden encontrar recursos tanto con licencia de pago como gratuitos por atribución. A continuación se le presenta al lector un listado de algunas páginas donde diversos desarrolladores de contenido publican su trabajo para que esté disponible al público bajo diferentes tipos de licencias.

Creative Commons
La página de Creative Commons contiene un apartado de proyectos con licencia CC. Éste apartado se encuentra dividido en diferentes categorías como la ciencia, la música y las artes.
http://creativecommons.org/

DeviantART
Página con recursos artísticos diversos. Se encuentra dividida por categorías y se pueden encontrar recursos tanto con licencia gratuita como de paga.
http://www.deviantart.com/

CGTextures
Página que contiene una amplia colección de imágenes y texturas en diferentes resoluciones categorizadas en más de 50 apartados.
http://www.cgtextures.com/

Capítulo 6: Análisis de la parte gráfica de la aplicación

La aplicación a desarrollar tiene fines de simulación y por lo tanto requiere del uso de elementos especializados como los que se encuentran en los proyectos de realidad virtual, es decir elementos que van más allá de los gráficos clásicos de una aplicación móvil y que permiten un grado de interactividad mayor. Para desarrollarlos las herramientas existentes son muy diversas pero el diseño e implementación de estos depende en gran medida de los requerimientos de la aplicación de manera que se logre un equilibrio entre funcionalidad, peso[1] y eficiencia.

A lo largo de éste capítulo se describen los requerimientos de la aplicación, algunas herramientas con las cuales se pueden cubrir dichos requerimientos y se selecciona aquella que los cumpla de la mejor manera con base en las características de dichas herramientas.

6.1 2D vs 3D

La forma de representar la información procesada en una aplicación puede llevarse a cabo de muy diversas maneras que pueden ir desde las más simples, como el despliegue de una o más líneas de texto, hasta formas visuales más complejas y llamativas. En el caso de los programas de simulación se utilizan con mayor frecuencia elementos gráficos con características representativas de aquél elemento que se pretende simular. Las dos formas más comunes de crear dichas representaciones es mediante elementos bidimensionales (2D) o tridimensionales (3D).

Un elemento bidimensional o 2D es aquél que necesita únicamente de dos dimensiones para representarse. Una fotografía digital es por ejemplo un elemento bidimensional, ya que puede representarse por infinidad de pixeles de colores en dos dimensiones: ancho y alto.

Un elemento tridimensional o 3D es aquél que requiere de tres dimensiones para su representación, por ejemplo anchura, altura y profundidad. Al tener una dimensión más que los 2D, los elementos 3D tienen la capacidad de representarse por medio de volúmenes, es decir, no se limitan a figuras planas como lo es en el caso de los primeros.

[1] Se refiere al tamaño en bytes del programa

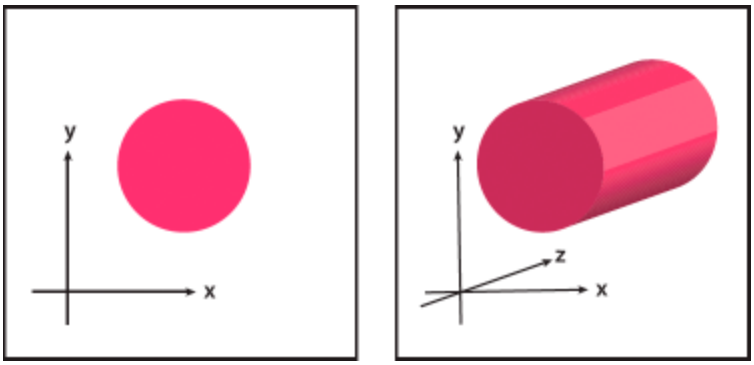


Figura 20: Objeto bidimensional a la izquierda y tridimensional a la derecha

Las interfaces de diferentes simuladores de circuitos utilizan elementos tridimensionales o bidimensionales para representar menús, componentes, herramientas o cualquier elemento disponible en ellos. Tomemos como ejemplo el programa Proteus; como se vio en el capítulo 2, el programa cuenta con dos módulos: ISIS, encargado del desarrollo de esquemas 2D y ARES, módulo con componentes 3D para la elaboración de circuitos impresos. Proteus tiene estos dos tipos de interfaz debido a que la finalidad de cada módulo es distinta y la forma de representar la información es más agradable e ilustrativa si se utilizan figuras 2D o 3D.

El escoger entre elementos 2D y 3D depende de diversos factores que están estrechamente ligados a los objetivos y al alcance de la aplicación en cuestión, a continuación se describen los factores más destacables de la aplicación desarrollada en esta tesis con base en los cuales se decidió la utilización u omisión de elementos bidimensionales o tridimensionales.

- Simbología de circuitos

En muchas áreas del conocimiento se utilizan símbolos para representar algún elemento o característica de dicha área, tomemos por ejemplo a la Química, en ella hay símbolos que representan cuando un producto es tóxico, flamable, corrosivo, etc. En el área de los circuitos eléctricos la simbología es muy importante, cada componente tiene un símbolo gráfico representativo cuya finalidad es brindar una forma sencilla de plasmar a la pieza en diagramas o planos de papel, es por esto que dichos símbolos son fácilmente representables por un elemento 2D.

Tomando como ejemplo uno de los diagramas encontrados en el libro[2] de Thomas L. Floyd podemos observar el uso de elementos gráficos bidimensionles encontrados con frecuencia en diagramas de circuitos eléctricos.

[2] Consultar la sección de bibliografía para información mas detallada

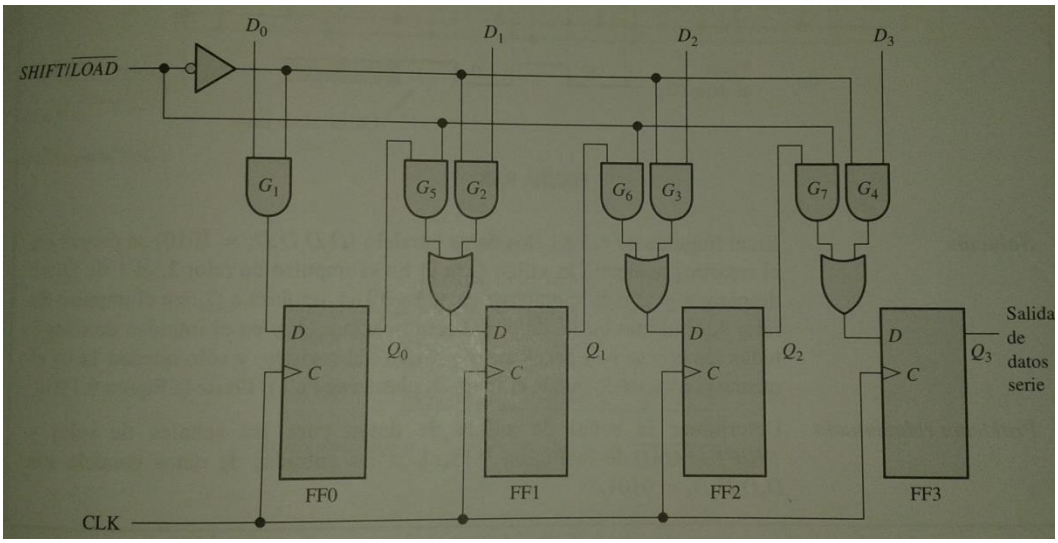


Figura 21: Uso de la simbología con elementos 2D

Algunas aplicaciones utilizan una representación 3D de la forma física de los circuitos, por ejemplo el módulo ARES de Proteus. Cabe mencionar que la forma física de un componente eléctrico varía enormemente de su representación simbólica, la mayoría de los componentes electrónicos están construidos mediante un encapsulado cuadrado con pines sobresalientes para las entradas y salidas de datos.

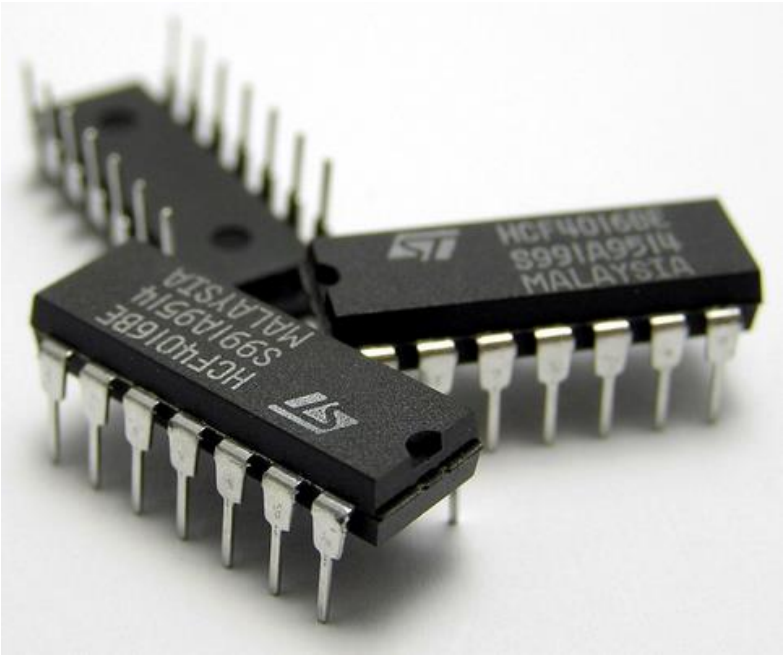


Figura 22: Ejemplo de encapsulado de circuitos integrados

A estos encapsulados se les conoce como *Circuitos Integrados* y son difíciles de diferenciar entre componentes ya que muchos llegan a tener la misma forma y un

mismo número de pines, la única manera de distinguirlos entre si es mediante un código clave característico de cada pieza impreso sobre el encapsulado.
El encapsulado es fácilmente representable por elementos 3D no obstante éste tipo de representación es muy poco efectiva para desplegar resultados en tiempo real ya que como se mencionó anteriormente, los componentes son fácilmente confundibles. Es por esto que para la aplicación de esta tesis se decidió utilizar simbología basada en elementos 2D fácilmente diferenciables y universalmente comprensibles.

A continuación se muestra una captura de pantalla de la aplicación con ejemplos de componentes en electrónicos que hacen uso de la simbología típica de los circuitos eléctricos.

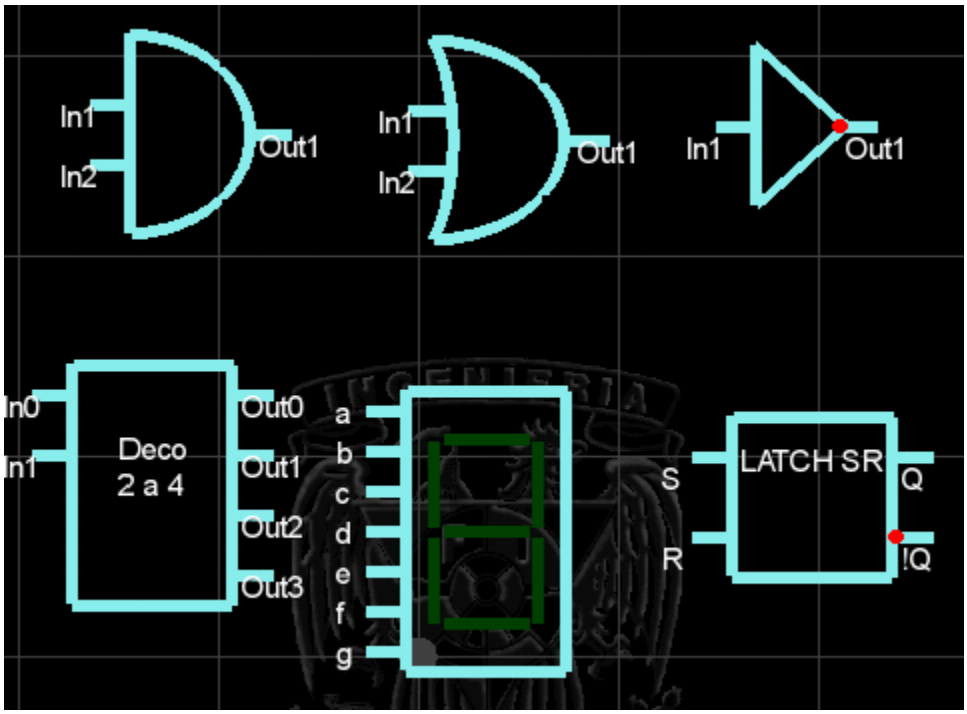


Figura 23: Ejemplo de simbología

- GUIs

Las interfaces de usuario o también conocidas como GUIs por las siglas en inglés de *Graphic User Interface* son elementos 2D que representan menús, botones y otras herramientas a las que puede acceder mediante la interacción del usuario con dispositivo periférico como un mouse o una pantalla táctil.

La finalidad de las GUIs es proveer al usuario de un programa una forma rápida, simple e intuitiva de realizar funciones utilizando elementos gráficos representativos afines a la operación o comando que ejecutan.

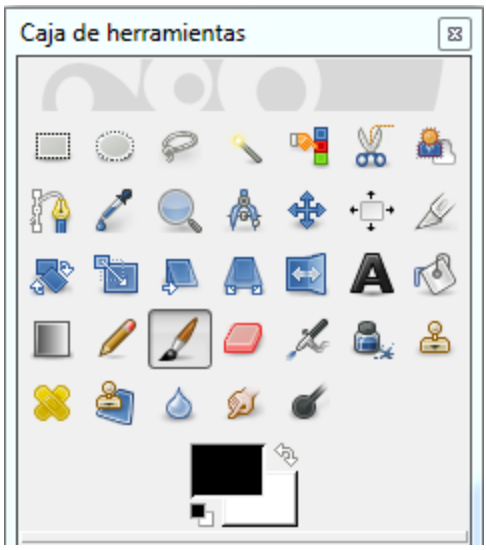


Figura 24: Ejemplo de GUI

Barra de herramientas de GIMP

La mayoría de las ocasiones las GUIs son elementos 2D debido a que, como se dijo anteriormente, su finalidad es meramente representativa y no requiere de mayor información o detalle del que puede ser proporcionado por un elemento bidimensional.
Para la aplicación de ésta tesis la interfaz de usuario es como la que se muestra a continuación.

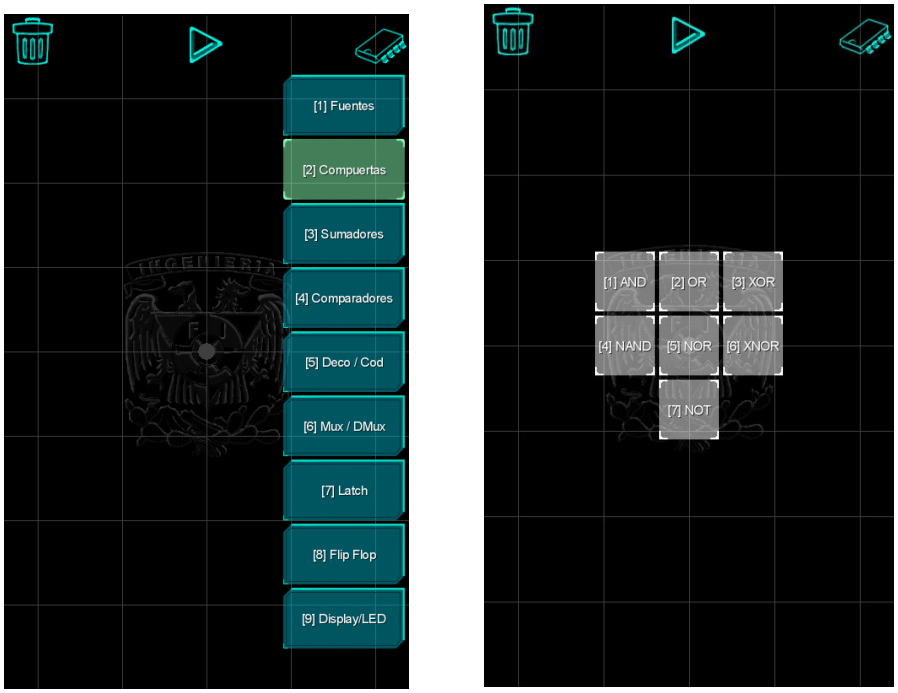


Figura 25: Interfaz de usuario de la aplicación

En ella los elementos gráficos ilustran su funcionalidad de una manera intuitiva para el usuario como en el caso de los botones de borrado, ejecución (Play) y de selección de componente ubicados en la parte superior de la pantalla, y de manera más explícita en los casos donde es necesario como se ve en el menú desplegable para selección de componente y en el sub menú (botones grises) de selección de pieza, los cuales describen respectivamente la familia de componentes y los componentes de dicha familia de manera textual.

Cabe mencionar que existen GUIs con elementos 3D pero estas son utilizadas muy ocasionalmente y con fines muy específicos. Un ejemplo de GUI tridimensional es el gizmo[3] de navegación 3D que podemos encontrar en programas como 3Ds Max o Unity.

Figura 26: Gizmo de Unity a la izquierda y de 3D Max a la derecha

Para la aplicación realizada en ésta tesis las GUIs 2D se ajustan perfectamente a los requerimientos de control de la aplicación ya que la simbología de los componentes electrónicos también es 2D y no hay elementos que requieran del tipo de manipulación compleja que brinda una GUI 3D.

- Interfaces de control (touch/mouse)

Una interfaz de control es todo aquel dispositivo o periférico que permite la manipulación de los elementos de un programa de una manera mucho más intuitiva para el usuario, ejemplos de éstos son el teclado y ratón de una computadora, la pantalla táctil de los dispositivos móviles o el control para un videojuego.
Una de las finalidades de la aplicación desarrollada en esta tesis es el despliegue en múltiples plataformas por lo que la forma de controlar los elementos del programa en ellas debe ser similar. Teniendo como primicia lo anterior, podemos tomar dos interfaces de control principales que se puede garantizar están presentes en la mayoría de los dispositivos sin importar la plataforma seleccionada, estas son el mouse y la pantalla táctil.

El mouse tiene la capacidad de ejecutar gestos muy similares a los que se pueden realizar con una pantalla táctil como se puede apreciar en la siguiente tabla comparativa:

[3] Control virtual interactivo capaz de controlar elementos de un programa

Gesto	Acción Mouse / Pantalla Táctil	Descripción
Press	Clic / Touch	Gesto que consta de 2 etapas, el clic/touch down y clic/touch up en la misma posición, es decir, no debe haber un arrastre (drag) entre dichas etapas.
Press Down	Clic mantenido / Touch mantenido	Etapa de un clic detectada una sola vez al momento en que es presionado un botón de mouse o el touch en una pantalla.
Press Up	Clic liberado / Touch liberado	Etapa de un clic detectada una sola vez al momento de soltar un botón de mouse o separar el dedo de una pantalla táctil.
Drag	Arrastre / Arrastre	Etapa intermedia entre un Press Down y un Press Up en donde el puntero del mouse o el dedo en una pantalla táctil se arrastra mientras se tenga presionado.

Tabla 2: Comparativa de gestos de Mouse y Touch

El uso de estas interfaces se ve apoyado de manera significativa por el aspecto de la implementación de elementos bidimensionales para representar la simbología de los componentes en la aplicación ya que dichos elementos son perfectamente acoplables a la manipulación mediante el uso de los gestos presentados en la tabla anterior.

6.2 Elección del Game Engine o framework

Para la construcción de la aplicación gráfica existe una gran variedad de herramientas disponibles que brindan todos los recursos necesarios para la programación y en algunos casos la compilación y ejecución de nuestra aplicación. A este tipo de herramientas dependiendo de su complejidad se les conoce como librerías, frameworks, software development kits (SDKs) o game engines, siendo el primero el

más simple y el último el más complejo.

De todos los anteriores las librerías son el elemento más básico que podemos utilizar. Las librerías son una colección de códigos que cumplen un fin específico, por ejemplo, se pueden programar librerías para dibujar líneas y otras para dibujar círculos. De esta manera, una vez que se tiene acceso a las librerías, el programador puede implementar el contenido de éstas y ahorrar valioso tiempo de programación reutilizando código ya existente y funcional.

Un framework como su nombre lo indica es un marco de trabajo que provee diversos tipos funcionalidad pre programada por medio de librerías y otros archivos. Estos se utilizan cuando se quiere tener una base sobre la cual comenzar a desarrollar y de esta manera aprovechar el tiempo que implicaría desarrollar todo desde cero. Los elementos de un framework se pueden complementar con código del usuario para modificar su comportamiento final.

Los SDKs también son un conjunto de herramientas que le facilitan al programador los archivos necesarios para desarrollar en una plataforma en específico, pero que también pueden incluir elementos de software y hardware más sofisticados como por ejemplo un IDE de programación o algún periférico necesario para utilizar el SDK. Un SDK puede contener varios frameworks.

Los Game Engines son sistemas de desarrollo mucho más completos a diferencia de los anteriores. Como su nombre lo sugiere están orientados al desarrollo de videojuegos pero también se llegan a utilizar para elaborar escenarios virtuales y proyectos relacionados con ambientes gráficos debido a la capacidad de las herramientas con las que cuentan. Los Game Engines ofrecen un entorno de desarrollo mucho más gráfico y amigable que le permite a cualquier usuario, incluso a aquel sin conocimientos técnicos, la elaboración de programas funcionales.
De todos los medios de desarrollo anteriormente descritos se seleccionaron tres, dos frameworks y un game engine, conocidos como Andengine, LibGDX y Unity 3D respectivamente, destacados por diferentes características entre las que se encuentran la popularidad de uso, documentación y el despliegue multiplataforma.

6.3 Game Engine vs framework

Para comprender mejor la elección de la herramienta utilizada en el desarrollo de la aplicación es importante comprender más a detalle las características de los dos tipos de herramienta seleccionados: el Game Engine y el Framework.

Framework es un término que se traduce como “marco de trabajo” y se refiere a un

conjunto de herramientas de programación que tienen como finalidad proveer una estructura base con la cual se puede dar solución a determinada problemática.

Pongamos un ejemplo para aterrizar de mejor manera el concepto. Supongamos que a un programador A se le asignó la tarea de desarrollar un programa que dibuje triángulos por medio de líneas y que a otro programador B se le encargó la elaboración de un programa que dibuje cuadrados también por medio líneas. Ambos programadores tienen que cubrir una tarea en común: crear un código que permita dibujar una línea y posteriormente utilizar dicho código para dibujar un conjunto de líneas y formar la figura correspondiente. Sabemos que cada persona tiene una forma de pensar única y por lo tanto una forma única de codificar, es por esto que no es de extrañarse que el código de despliegue de líneas del programador A sea diferente al del programador B, pero a final de cuentas ambos programadores habrán logrado desarrollar el programa que les fué encargado.

¿Qué pasaría si a un programador C se le asigna la tarea de elaborar una aplicación que dibuje cualquier figura por medio de líneas pero utilizando los códigos de los programadores A y B? C tendría que crear una forma propia de dibujar líneas para sustituir la de A y B ya que no tiene caso tener dos códigos que cumplen un mismo fin y que pueden llegar a ser incompatibles. Todo lo anterior se traduce en tiempo de desarrollo desperdiciado que podría haberse aprovechado eficientemente si se tuviera un marco de trabajo en donde se defina cómo construir líneas o en el que se brinden herramientas que las construyan automáticamente. Éste marco de trabajo es el que se conoce más comúnmente como framework en términos de software y es el que evita que se tenga que “reinventar la rueda” a cada momento.

Algunos frameworks pueden contener las siguientes herramientas:

- Bibliotecas de código y ejemplos.
- Soporte para uso con otros programas.
- Software de integración de recursos.

Se pueden encontrar muchos y diferentes frameworks que son útiles para un tema en particular pero pueden llegar a variar en aspectos que van de los más simples como la facilidad de implementación hasta los más complejos como la eficiencia.

Un *game engine* o *motor de juegos* es un sistema completo para el desarrollo de videojuegos y aplicaciones gráficas en general. Estos proveen un framework para la construcción de los programas con el fin de reducir el tiempo de desarrollo.
Los game engines de mejor reputación poseen módulos especializados encargados del cálculo de diferentes características comunes en las aplicaciones gráficas interactivas como por ejemplo el renderizado, el manejo de colisiones, sonidos, etc. Un game

engine es considerado como tal cuando posee la mayoría de los módulos enlistados a continuación:

- Programación
- Motor de gráficos o de renderizado (2D, 3D o ambos)
- Motor de física
- Colisionadores
- Audio
- Inteligencia artificial
- Conectividad

El objetivo principal de un game engine sigue el mismo principio que el de un framework, evitar que el desarrollador pierda tiempo “reinventando la rueda”, es decir, brindan las herramientas necesarias que facilitan el trabajo mediante elementos ya integrados o en su caso pre programados, de tal manera que el desarrollador pueda enfocar su tiempo en la implementación de un correcto comportamiento de su programa en lugar de enfocarlo en la resolución de funciones básicas.

La diferencia principal con un framework radica en que las herramientas que encontramos en un game engine suelen estar integradas en una sola instancia de programa lo que lo hace mucho más simple de utilizar ya que no requiere de la integración de diferentes archivos generados por programas ajenos para conseguir un producto final, sin embargo cabe mencionar que existen algunos tipos de game engines que limitan sus herramientas a las estrictamente esenciales, por ejemplo los que ofrecen únicamente la función de renderizado, a estos se les conoce como *graphics engine* o *motores de gráficos* y se caracterizan por tener la capacidad de renderizar gráficos exclusivamente, ya sea 2D o 3D, por lo que habría que utilizar otros programas para cubrir los demás requerimientos de la aplicación que se esté elaborando con éste tipo de engine.

Las ventajas de utilizar un game engine son numerosas, entre las más importantes destacan la optimización del tiempo de desarrollo, la disponibilidad de herramientas precargadas y la portabilidad, es decir, la facilidad de migrar el proyecto final entre plataformas sin necesidad de hacer cambios significativos al programa. Sin embargo existe cierta mal interpretación alrededor término game engine y es que por todas las características que éste ofrece se llega a pensar que con él se pueden elaborar todos los recursos necesarios para un proyecto. Esto no es del todo cierto, la mayoría de los engines funcionan más como herramientas de integración de recursos que de generación de los mismos y es que aunque algunos de ellos ofrecen herramientas capaces de generar ciertos tipos de recursos para la aplicación, la mayoría deberán ser creados previamente mediante el uso de software especializado. Por ejemplo, si se

tiene pensado elaborar un juego 3D de carreras de autos primero hay que generar los modelos de los autos, el game engine no es capaz de construirlos pero existe software de modelado como 3Ds Max o Blender que son capaces generar la geometría necesaria para modelar un auto virtual; los modelos a su vez necesitan texturas para el aspecto visual, las cuales se generan con software de edición de imágenes como Photoshop o GIMP. Una vez que se tiene el modelo del auto construido y texturizado se puede agregar al engine como un recurso o *asset* para que se le programe el comportamiento deseado.

Ahora que se tiene una mejor idea de los términos Framework y Game Engine se procede a definir las características de cada una de las herramientas de manera individual para posteriormente describir la decisión final.

6.3.1 Unity

Unity es un entorno de desarrollo de juegos y aplicaciones gráficas interactivas tanto 2D como 3D. La característica más sobresaliente de éste engine es que utiliza una interfaz gráfica con elementos intuitivos para la construcción de sus aplicaciones con lo cual se ahorra mucho tiempo de desarrollo pero además permite que cualquier persona con conocimientos técnicos mínimos pueda aprender a desarrollar una aplicación con poco esfuerzo.
Está disponible en versiones para los sistemas operativos de escritorio más populares del mercado: Windows, Linux y Mac y se puede obtener por descarga directa desde su página oficial:

unity http://unity3d.com/	

Unity es un software comercial, sin embargo cuenta con dos tipos de licencia: Unity Pro y Unity Free, siendo la segunda una licencia gratuita con funciones más limitadas.

Se puede encontrar un listado de las diferencias entre licencias en la sección de

licencias de la página oficial de Unity en la siguiente dirección:

http://unity3d.com/unity/licenses	

Como todo game engine, cuenta con diferentes módulos especializados que facilitan el desarrollo de una aplicación gráfica. Los módulos de Unity son los siguientes:

- Motor de renderizado
- Motor de física
- Módulo de Audio
- Módulo de animación
- Módulo de programación
- Módulo de red
- Soporte multiplataforma

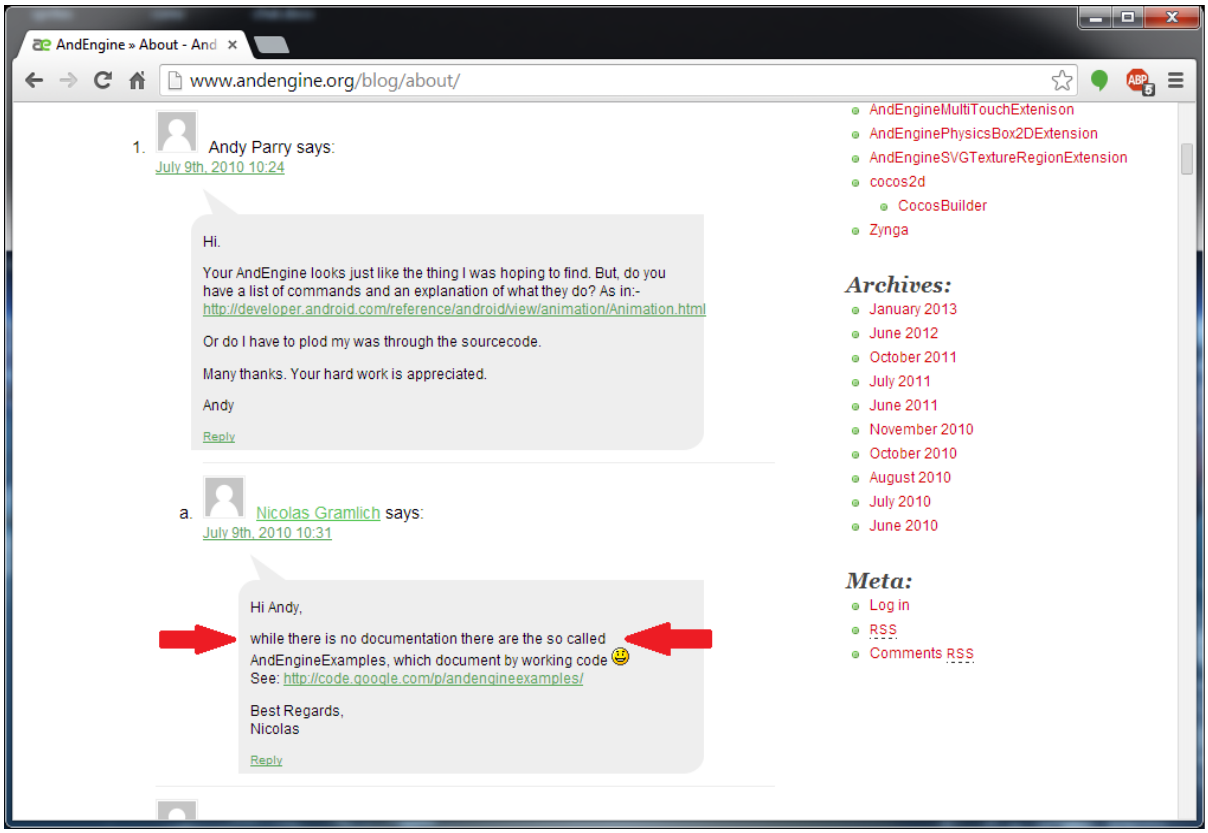
6.3.2 AndEngine

AndEngine es un framework de código abierto que permite la creación de aplicaciones 2D mediante el uso de *OpenGL*, otro framework de gráficos codificado en lenguaje C.
Las aplicaciones creadas con AndEngine son ejecutables exclusivamente en el sistema operativo móvil Android pero cuentan con la ventaja de ofrecer una abstracción de código mucho más propicia para el desarrollo de gráficos que la del código nativo de dicha plataforma.
Éste framework cuenta con los elementos necesarios para cubrir los aspectos más importantes en el desarrollo de una aplicación gráfica interactiva como lo es el renderizado de gráficos bidimensionales, adaptación de resolución, módulo de audio, colisiones, administración de datos y un ciclo de vida de aplicación que se adapta a los diferentes estados del sistema operativo Android.
Como muchos otros frameworks, requiere de un IDE externo para la elaboración de un proyecto ya que éste no cuenta con una interfaz propia de programación. El IDE más comúnmente utilizado junto con este framework es Eclipse, un entorno de desarrollo

también de código abierto descrito en el capítulo 4 de ésta tesis.
Los archivos con los recursos de AndEngine están disponibles para su descarga en la siguiente dirección:

https://github.com/nicolasgramlich/AndEngine	

Cabe mencionar que éste framework sufre de uno de los males comunes del código abierto: la documentación, y es que a diferencia de otros frameworks de éste tipo, AndEngine carece de un API de documentación y de una página oficial de calidad a la cual referenciarse, incluso el mismo desarrollador del framework, Nicolas Gramlich, admite la inexistencia de documentación al contestar un comentario en el blog de AndEngine y menciona que se pueden encontrar ejemplos gratuitos.



Nicolas Gramlich, desarrollador de AndEngine afirmando la falta de documentación del framework en un comentario del blog del mismo.

Existe un libro de la editorial *Packt publishing* en el que se describe con mas detalle la estructura de este framework y que puede cubrir la falta de documentación en la web. La ficha bibliográfica de éste libro se encuentra a continuación.

Schroeder, Jayme, and Brian Broyles. *AndEngine for Android Game Development Cookbook: Over 70 Highly Effective Recipes with Real-world Examples to Get to Grips with the Powerful Capabilities of AndEngine and GLES2*. Birmingham: Packt Publ., 2013.

La página oficial de AndEngine se encuentra en la siguiente dirección:

 http://www.andengine.org/	

6.3.3 LibGDX

LibGDX es un framework de código abierto, multiplataforma, basado en Java que permite la creación de aplicaciones gráficas interactivas tanto 2D como 3D haciendo uso de una versión de OpenGL conocida como OpenGL ES.

Su principal característica es la de brindar un marco de trabajo en el que se puede desarrollar, ejecutar y depurar un proyecto en plataformas de escritorio de forma nativa, es decir la aplicación se ejecuta dentro de una ventana como un programa nativo del sistema operativo, de esta manera se omite despliegue forzoso en la plataforma objetivo o el uso de emuladores para probar cada cambio que se haga en el programa durante el proceso de construcción, garantizando además el mismo comportamiento.

LibGDX cuenta con módulos que brindan todo lo necesario para la construcción de una aplicación gráfica como módulo de renderizado, audio, física, fórmulas matemáticas y de administración de archivos para lectura y escritura de información. Cada módulo se encuentra estructurado en clases de manera que su uso es opcional y adaptable a los requerimientos del proyecto, esto permite que la aplicación final sea mucho más ligera y eficiente al utilizar solamente los recursos necesarios.

Las plataformas soportadas por este framework son los sistemas operativos de escritorio Windows, Linux y Mac y los sistemas operativos móviles Android, iOS y BlackBerry, además permite el despliegue del programa como aplicación web para navegadores de Internet.

El framework se puede descargar en un archivo comprimido desde la página oficial de LibGDX en la siguiente dirección:

libGDX http://libgdx.badlogicgames.com/index.html	

El archivo comprimido además de contener las librerías que componen los módulos del framework contiene herramientas de desarrollo útiles para la creación o edición de recursos varios para una aplicación. Estas herramientas son las siguientes:

- **Gdx Setup UI**

  Evita el proceso de configuración manual mediante una interfaz gráfica que se encarga de crear los proyectos para el IDE Eclipse ya configurados con los archivos necesarios de LibGDX.

- **Particle editor**

  Interfaz gráfica para la edición de efectos de partículas en tiempo real. Permite crear efectos de partículas a partir de una imagen y la modificación de variables características de las mismas como el tiempo de vida, velocidad, escala, etc.

- **Texture packer**

  Interfaz gráfica que permite el empaquetado de todas las imágenes utilizadas en un proyecto en una sola conocida como *atlas* para optimizar el proceso de carga de las mismas.

- **Bitmap font generator**

  Herramienta de creación de tipografías tipo mapa de bits (.BMF) a partir de tipografías tipo True Type Font (.TTF).

LibGDX cuenta con muy buena documentación, en su página oficial se puede encontrar un listado de enlaces a diferentes sitios web donde los autores del framework, Mario Zechner y Nathan Sweet, así como diferentes usuarios del mismo, ofrecen tutoriales escritos y en video. Todo lo anterior se puede encontrar en la siguiente dirección:

http://libgdx.badlogicgames.com/documentation.html	

6.4 Parámetros a considerar

Antes de seleccionar alguno de los freworks o el engine anteriormente descritos como herramienta de desarrollo para la aplicación de esta tesis, hay tres parámetros de gran relevancia para el programa que se deben considerar al momento de tomar la decisión definitiva, las licencias, la documentación y el código genérico. A continuación se describe cada uno de estos parámetros a detalle.

6.4.1 Licencias

La licencia del framework o engine es fundamental para futuras versiones de la aplicación, tanto una licencia de código abierto como una licencia comercial abren la posibilidad de un crecimiento a futuro, sin embargo, dentro de ese crecimiento se plantea la posibilidad de lucrar con la aplicación y dicha posibilidad se puede ver limitada con una licencia comercial ya que el costo de ésta no es algo que sea fácilmente financiable por un desarrollador independiente.

Tomemos como ejemplo el costo de la licencia de Unity, al día de hoy, *Unity Pro* está disponible por 1500 dólares, esto es casi 20,000 pesos mexicanos, esto sin considerar los “aditivos” para las plataformas móviles cuyo precio es igualmente 1500 dólares cada uno.

Siguiendo con el caso específico de Unity, recordemos que anteriormente en éste capítulo se menciona que el engine cuenta con una licencia gratuita que entre otras cosas ofrece la posibilidad de lucrar con las aplicaciones creadas siempre que la aplicación sea para plataformas móviles y se sea desarrollador independiente o un estudio pequeño o en desarrollo, siendo éstos más específicamente todas aquellas entidades cuyos ingresos brutos anuales no superen los 100 000 dólares, es decir, los $1 325 416.18 pesos mexicanos.

Con cualquiera de éstos dos tipos de licencia la aplicación de esta tesis se ve limitada

en diferentes objetivos. En el caso de uso de una licencia comercial el precio de la misma rompería con el objetivo del abatimiento de costos de desarrollo. Por otro lado, la licencia gratuita está limitada a la publicación sin restricciones en plataformas móviles, no así en las de escritorio; esto limita a la aplicación en cuanto al despliegue en plataformas de escritorio.

Pasemos ahora al escenario de las licencias de código abierto. Tanto AndEngine como LibGDX son frameworks que ofrecen licencias de software libre con las que se pueden realizar aplicaciones con o sin fines lucrativos.

AndEngine se encuentra bajo una licencia GNU LGPL, siglas de *GNU Lesser General Public License,* licencia creada por la Free Software Foundation y que defiende las libertades del software libre. La GNU LGPL es similar la licencia GNU GPL, o bien *GNU General Public Licence* que estipula que todo aquél elemento registrado con esta licencia tiene la capacidad de usarse, compartirse, estudiarse o modificarse libremente. Ésta licencia (GPL) fue creada por Richard Stallman y es de tipo *CopyLeft* que implica que todo trabajo derivado solamente podrá ser distribuido bajo el mismo tipo de licencia.
La diferencia principal entre GNU GPL y GNU LGPL es que ésta última permite mezclarse con trabajos registrados con licencias no-GPL o no-LGPL ya sean de software libre o propietario, sin embargo al integrarse con éstos las áreas disponibles para el ejercicio de los derechos de software libre serán únicamente las que se encuentren al alcance de los términos de la GNU LGPL.

La palabra en inglés “Leeser”, al principio del nombre de la GNU LGPL, significa “menor” en español y se utiliza para resaltar el hecho de que ésta licencia no garantiza al usuario final una libertad completa como lo haría la GNU GPL.

LibGDX se encuentra bajo una licencia Apache 2.0, ésta es una licencia de software libre creada por la *Apache Software Foundation (ASF)* que, al igual que otras licencias de software libre, garantiza la libertad de uso del software con cualquier propósito, de modificar y distribuir versiones modificadas del software sin tener que cubrir regalías.

La diferencia de la licencia Apache con otras licencias similares radica en que con ésta no se requiere que el trabajo derivado sea licenciado bajo los mismos términos, incluso ni siquiera como software libre, sin embargo se requiere de mantener el mismo tipo de licencia en todas aquellas partes que se preservan íntegras del trabajo original.

Con cualquiera de las licencias anteriores la aplicación puede crecer de la forma deseada contando además con la ventaja del abatimiento de costos por lo que el uso de cualquiera de los dos frameworks se considera una muy buena opción para el

desarrollo de la aplicación de ésta tesis.

En conclusión, en cuanto a licenciamiento, es recomendable utilizar los frameworks de desarrollo ya que manejan una licencia de código abierto que no se contrapone a los objetivos definidos para la aplicación de esta tesis.

6.4.2 Documentación

En el área del software es una muy buena práctica el documentar el código que se esté generando con el fin de facilitar a otras personas la inteligibilidad del mismo, de ésta manera si el código requiere ser modificado por personas ajenas a los autores originales el proceso pueda llevarse a cabo de una manera mucho más eficiente al no tener que estudiar el comportamiento de cada una de las líneas que lo conforman.
En el caso de los frameworks y engines la documentación es igualmente importante ya que ésta define en gran parte la facilidad de uso de los mismos.

A la documentación de una herramienta de programación se le conoce comúnmente como Interfaz de Programación de Aplicaciones o *API* por las siglas en inglés de *Application Programming Interface*. En ella se define la forma en que los diferentes elementos de una herramienta de software pueden ser utilizados en una aplicación.

Unity cuenta con una API en línea conocida como *Script Reference*, en ella se describen cada una de las clases de programación que conforman a éste engine y que pueden ser utilizadas en un script de Unity para definir algún comportamiento en un proyecto. Se puede consultar en la siguiente dirección:

https://docs.unity3d.com/Documentation/ScriptReference/	

Por ser un software comercial, la información disponible en el script reference es muy completa, cuenta con descripciones escritas del comportamiento de las diferentes clases que conforman los elementos de Unity y además cada una de ellas contiene un

listado de métodos, variables y ejemplos de código en los diferentes lenguajes soportados por el engine.

El framework LibGDX también cuenta con una API muy completa. Al estar basado en Java la documentación del framework se realiza de la misma manera que en dicho lenguaje de programación, esto es mediante una herramienta conocida como *Javadoc*. Javadoc es un analizador sintáctico, reconocido como estándar para la documentación de clases de Java, que detecta etiquetas de texto precedidas por el símbolo "@" y que genera un conjunto de páginas en formato HTML donde se despliegan las descripciones de las clases, métodos y atributos en los cuales se hayan utilizado dichas etiquetas.

La mayoría de los IDEs de desarrollo que soportan el lenguaje Java integran la herramienta Javadocs para su uso.

En la siguiente dirección se puede consultar la API de LibGDX:

http://libgdx.badlogicgames.com/nightlies/docs/api/	

LibGDX por lo tanto se considera un elemento recomendable en el desarrollo de la aplicación de ésta tesis.

AndEngine es un framework que carece de documentación. A pesar de estar basado en Java no cuenta con documentación generada mediante Javadocs ni existe hasta la fecha una página en internet u otro recurso en línea que de manera oficial de soporte a este framework y que brinde al usuario la facilidad de referencia y consulta de los elementos que lo constituyen.

En conclusión, en cuanto a documentación se trata, tanto Unity como LibGDX se consideran muy buenas herramientas de desarrollo para la aplicación de esta tesis debido a que cuentan con APIs muy bien estructuradas. AndEngine sin embargo no se considera un elemento recomendable ya que la falta de documentación complica la

implementación del código y por lo tanto ralentiza el desarrollo.

6.4.3 Código genérico

El código genérico es aquel que ofrecen los frameworks y engines de desarrollo mediante el cual se pueden programar aplicaciones para diferentes plataformas sin necesidad de utilizar el código nativo de cada una de ellas. Su principal ventaja radica en la implementación de un código único compatible con todas los sistemas operativos soportados por el framework o engine que se esté utilizando, de ésta manera la lógica del programa se escribe una sola vez en vez y el resultado final es el mismo[4] independientemente del dispositivo en el que se ejecute la aplicación, así mismo se evita el proceso de exportación o recodificación de la lógica del programa en el lenguaje nativo de cada plataforma.

Unity posee librerías con código genérico en tres diferentes lenguajes de programación: JavaScript, C# y Boo. Además en éste engine en específico se cuenta con la característica adicional de la interoperabilidad entre scripts independientemente del lenguaje en el que estén escritos. Todo esto hace de Unity un engine que garantiza un desarrollo simplificado y en un tiempo considerablemente corto, sin embargo para la aplicación de ésta tesis la integración de todos los elementos en Unity llega a ser un problema que se refleja en el peso de la aplicación y en el aprovechamiento de recursos. En Unity una aplicación vacía, es decir sin objetos virtuales ni scripts de interactividad, animaciones, modelos 3D, etc, simplemente una cámara, llega a pesar aproximadamente 9MB (megabytes) como mínimo y a partir de esa medida el peso de la aplicación comienza a crecer dependiendo de los assets[5] que contenga. Esto se debe a que Unity integra todos los recursos que considera como mínimos necesarios para una aplicación y su despliegue en cierta plataforma.

Si tomamos como referencia a la aplicación Every Circuit se puede observar en la imagen siguiente que el tamaño de ésta es de aproximadamente 11MB por lo que la aplicación de ésta tesis tendría un margen de 2MB para integrar todos los recursos necesarios que permitan lograr una funcionalidad equiparable. Esto es en extremo difícil de conseguir por lo que a pesar de ser un engine con muchas bondades, Unity no es una herramienta óptima para el desarrollo de la aplicación.

[4] El resultado puede llegar a variar en cuanto a las interfaces de hardware con las que cuente cada dispositivo para el control de la aplicación.

[5] Se refiere a todos aquellos recursos útiles para el desarrollo de una aplicación como imágenes, música, modelos 3d entre otros.

Figura 27: Información de EveryCircuit Free

LibGDX también es un framework que cuenta con código genérico para la programación de aplicaciones. La estructura del framework está diseñada de manera tal que permite dividir un proyecto en subproyectos que contienen la lógica principal de la aplicación y un lanzador[6] por cada una de las plataformas objetivo.

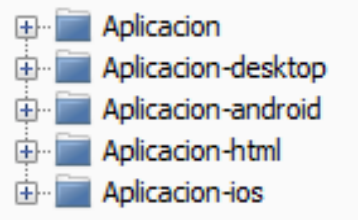


Figura 28: Vista previa de la estructura de un proyecto creado con LibGDX.

El primer proyecto de arriba hacia abajo contiene la lógica del programa, los proyectos precedidos por un guión y el nombre de la plataforma son los lanzadores de la aplicación para cada una de éstas.

Mediante ésta estructura LibGDX brinda al programador la posibilidad de utilizar las librerías y herramientas estrictamente necesarias para la construcción del proyecto y su

[6] Se refiere a un proyecto secundario que instancia los componentes del proyecto principal que contiene la lógica de la aplicación y que despliega el programa en la plataforma a la que esté orientado el lanzador.

despliegue en cada plataforma, logrando así un menor tamaño en la aplicación final.

LibGDX aprovecha al máximo su código genérico permitiendo la depuración del programa en tiempo de ejecución utilizando un despliegue nativo en plataformas de escritorio, evitando así el uso de emuladores y dispositivos físicos o la compilación del proyecto entero cada que se ejecuta una prueba.

LibGDX se considera una opción viable para el desarrollo de la aplicación de esta tesis ya que además de contar con todas las características antes descritas permite un control de los elementos del programa a un nivel más profundo que no compromete el tamaño de la aplicación y ofrece una amplia variedad de sistemas operativos soportados.

AndEngine a pesar de tener un código distinto al código nativo de Android se descarta por anticipado como posible herramienta de desarrollo para la aplicación de ésta tesis ya que como se mencionó anteriormente es un framework que permite crear aplicaciones exclusivamente para el sistema operativo móvil Android y en consecuencia interfiere con uno de los objetivos de la misma: el despliegue multiplataforma.

6.5 Selección final del game engine

Para determinar la herramienta final a utilizar en el desarrollo de la aplicación se construyó la siguiente tabla que califica cada framework y engine candidato con base en las conclusiones obtenidas en los subtemas anteriores de este capítulo.

	Licencia	Documentación	Código genérico
Unity	✗	✓	✗
AndEngine	✓	✗	✗
LibGDX	✓	✓	✓

Figura 29: Calificación de herramientas de desarrollo

De la tabla se observa que el único elemento que cuenta con una nota positiva en todos sus rubros es LibGDX por lo tanto se seleccionó a éste como el herramienta de desarrollo que trabaje en conjunto con las herramientas de diseño y programación seleccionadas en la elaboración de la aplicación de esta tesis.

Para concluir el capítulo es importante mencionar que el hecho de que unos u otros engines o frameworks hayan sido seleccionados no quiere decir que los demás sean malos, por el contrario todos tienen una ventaja dependiendo de los requerimientos del proyecto a construir.

Capítulo 7: Despliegue multiplataforma

El despliegue multiplataforma se refiere a la característica de los programas computacionales que les permite ejecutarse en diferentes sistemas operativos habiendo sido codificados mediante un código genérico.

Las ventajas derivadas de éste tipo de despliegue se ven reflejadas desde el momento de la programación de la aplicación ya que como se dijo anteriormente, el código genérico evita la implementación de código nativo, pero además de esto, hablando en términos de distribución, la aplicación se beneficia con muchas más probabilidades de expansión al no verse limitada a la disponibilidad de un solo marketplace[7].

Para la aplicación de esta tesis el despliegue multiplataforma impulsa en primera instancia el alcance la herramienta, es decir, permite que la aplicación pueda ser utilizada por el mayor número de personas posible sin importar el tipo de dispositivo con el que se cuente. De esta manera los beneficios ofrecidos en la aplicación tendrán a su vez un mayor alcance en un número elevado de usuarios. Además, este tipo de despliegue se presenta como una característica adicional que no se encuentra en muchos programas de simulación existentes, por ejemplo Proteus y Xilinx son programas exclusivos para computadoras de escritorio mientras que Every Circuit, a pesar de ser una aplicación móvil, se encuentra únicamente disponible para el sistema operativo Android.

7.1 Plataformas objetivo

Para darnos una idea más clara del alcance que puede tener una aplicación multiplataforma observemos las siguientes estadísticas basadas en la información del sitio NetMarketshare con un rango de medición de Enero a Marzo del 2014.
Se deja además al lector la referencia del sitio en la siguiente dirección.

NETMARKETSHARE Market Share Statistics for Internet Technologies http://www.netmarketshare.com/	

[7] Tiendas de aplicaciones de los sistemas operativos del mercado mediante las cuales se puede adquirir software para un dispositivo móvil o computadora.

La siguiente gráfica muestra la distribución de diferentes sistemas operativos para dispositivos móviles de acuerdo a su aceptación en el mercado.

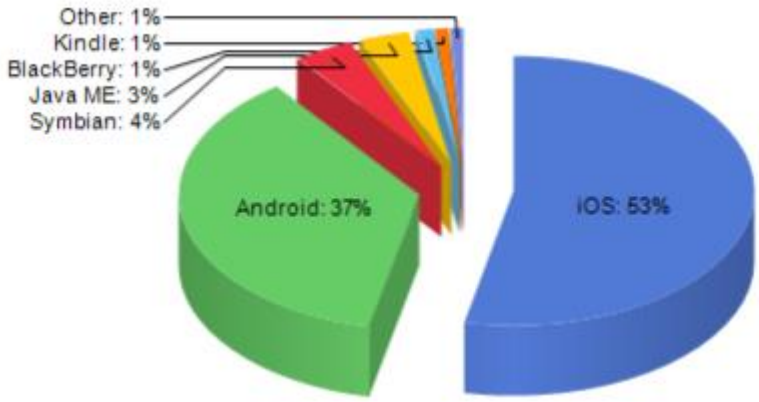


Figura 30: Distribución de los S.O. móviles más usados de Enero a Marzo del 2014 de acuerdo a NETMARKETSHARE

De la gráfica se puede observar que hay dos sistemas operativos que abarcan el 90% del mercado: iOS y Android, esto quiere decir que el realizar una aplicación ejecutable en estas dos plataformas como mínimo garantiza una cobertura casi total en el terreno móvil y en consecuencia una mejor expansión.

Veamos ahora la distribución que tienen los sistemas operativos de escritorio.

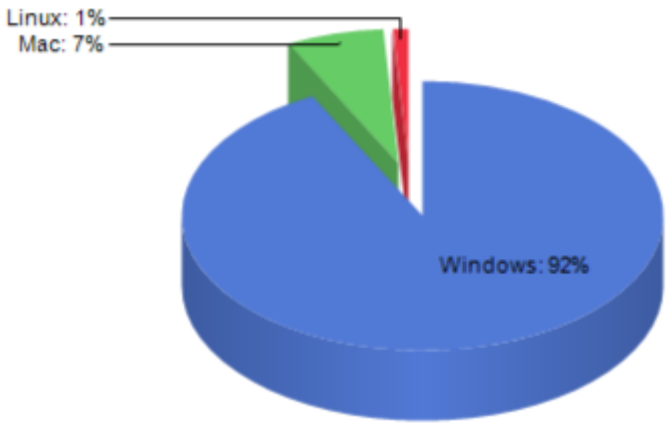


Figura 31: Distribución de los S.O. más usados de Enero a Marzo del 2014 de acuerdo a NETMARKETSHARE

Al igual que en la primera gráfica podemos apreciar que Windows es quien tiene mayor influencia en el mercado y por lo tanto hacer una aplicación compatible con éste sistema operativo garantiza una muy buena expansión.

Con base en las gráficas anteriores las posibles plataformas objetivo para la aplicación de ésta tesis serían las dos mejor aceptadas tanto para sistemas operativos móviles como de escritorio, estas son: iOS, Android, Windows y Mac, sin embargo queda un parámetro a considerar antes de declarar una selección final, el licenciamiento de cada plataforma.

7.2 Licencias de plataforma

Existen ciertas plataformas que requieren un licenciamiento forzoso de la aplicación que permita su distribución y su uso. A continuación se presenta al lector el tipo de licenciamiento requerido para la aplicación de ésta tesis en específico de acuerdo a cada uno de los sistemas operativos contemplados como plataformas objetivo.
De acuerdo a los requerimientos de las licencias descritas a continuación se definirán las plataformas objetivo finales.

7.2.1 Sistemas operativos móviles

En el caso de las plataformas móviles el licenciamiento es forzoso ya que su distribución dentro de los marketplaces requiere de una licencia que identifique al autor o autores de la aplicación como desarrolladores oficiales en la plataforma en cuestión.

7.2.1.1Android

Para licenciar una aplicación para el sistema operativo Android se deben seguir los pasos descritos a continuación:

1. Crear una cuenta de Google.
   Se debe tener una cuenta de Google, puede ser aquella con la que hacemos uso de alguno de los servicios de Google como Gmail, Google Play, YouTube, etc.
   En la siguiente dirección se puede obtener una cuenta de forma gratuita.

Google https://accounts.google.com/	

2. Darse de cómo desarrollador y pagar una cuota.
   Con la cuenta de Google se debe acceder a la consola de desarrollo de Google Play para dar de alta la cuenta como cuenta de desarrollador y realizar el pago de la cuota de 25 dólares. Esta cuota es de pago único, es decir, una vez que se paga no hay necesidad de renovarla cada cierto tiempo, te autoriza como desarrollador de por vida.
   La consola de desarrollo de Google Play se encuentra en la siguiente dirección:

Google play https://play.google.com/apps/publish/	

3. Darse de alta en Google Merchant (opcional).
   Si se quiere realizar la venta de aplicaciones o de algún otro tipo de contenido dentro de Google Play se requiere dar de alta la cuenta en Google Merchant, esto se realiza en la misma dirección del paso 2 dentro de la sección **Financial reports** , **Setup a Merchant Account now.**

7.2.1.2 iOS

Para licenciar una aplicación para iOS se deben seguir los pasos descritos a continuación:

1. Registrarse como Apple Developer (desarrollador de 57ersi).
   Este registro se realiza desde la página de registro de Apple Developer mediante un *Apple ID*. Un Apple ID es una cuenta de usuario que le permite a una persona hacer uso de todos los productos y servicios de la empresa Apple como compra de aplicaciones en la App Store, la descarga de música y películas en iTunes, etc. Se puede realizar el registro como Apple Developer y encontrar un enlace para obtener un Apple ID en la siguiente dirección, ambas de manera gratuita.

Developer https://developer.apple.com/register/index.action	

Como se mencionó anteriormente, el registro como Apple Developer se puede realizar mediante un Apple ID y garantiza el acceso a las siguientes herramientas de desarrollo:

- Documentación de iOS en sus versiones estables (no beta).
- Herramientas de desarrollo.
- Prueba de aplicaciones exclusivamente en el simulador de Xcode[8].

2. Unirse a un programa de desarrollador.
   Un programa de desarrollador permite crear aplicaciones bajo ciertos términos que varían de acuerdo al programa seleccionado. En otras palabras, elegir un programa de desarrollador es el equivalente a seleccionar un tipo de licencia para las aplicaciones que se desarrollen.
   Los programas de desarrollo para iOS se enlistan a continuación junto con una breve descripción de los mismos.
   - iOS University Program
     Este programa es gratuito para algunas universidades que tengan convenio, permite crear grupos de desarrollo de hasta 200 integrantes, las aplicaciones desarrolladas deberán ser no lucrativas, incluye acceso a herramientas de desarrollo como el SDK y otros recursos del iOS Dev Center[9], permite el despliegue en dispositivos físicos y el compartimiento de la aplicación entre los integrantes del grupo de desarrollo vía email o vía sitio web.

   - iOS Developer Program
     Garantiza el acceso a documentación y herramientas de desarrollo para versiones estables y beta del sistema operativo así como el despliegue en

[8] IDE de programación utilizado para la creación de aplicaciones de Apple
[9] Página web de Apple con recursos para desarrolladores
https://developer.apple.com/devcenter/ios/index.action

dispositivos físicos y la publicación de la aplicación el el App Store.
Tiene un costo de 99 dólares anuales.

- iOS Developer Enterprise program
  Permite desarrollar aplicaciones para uso privado dentro de una empresa permitiendo la instalación de la aplicación únicamente entre los empleados, además permite el despliegue en dispositivos físicos, soporte técnico a nivel de código y acceso al foro de Apple Developer.
  Tiene un costo de 299 dólares anuales.

7.2.2 Sistemas operativos de escritorio

El licenciamiento en los sistemas operativos de escritorio no es un factor indispensable para la aplicación de ésta tesis en específico debido a que como se describe en el capítulo 6 la herramienta utilizada para el desarrollo de la aplicación de código abierto y está basada en Java que también es un lenguaje de desarrollo gratuito. Esto quiere decir que la aplicación será compatible con todo aquél sistema operativo que sea compatible con Java. A continuación se describen los requerimientos de Java 59ersión 7 (JDK 7 y JRE 7) para cada uno de los 3 sistemas operativos de escritorio más utilizados.

7.2.2.1 Windows

En Windows Java está disponible para casi todas las versiones excepto para aquellas en versión móvil como por ejemplo Windows 8 en tablets.
En general, una máquina con una versión de Windows debe cumplir con los siguientes requerimientos mínimos:

- RAM: 128 MB (64 para Windows XP)
- 124 MB de espacio en disco.

Cabe mencionar que en el caso específico de Windows XP Java no considera este sistema operativo como una plataforma compatible con versiones de Java liberadas después del 8 de Abril del 2014 ya que a partir de esta fecha Microsoft dejó de dar soporte de manera oficial a dicha versión de Windows.

7.2.2.2 MacOS

- Mac basada en Intel con Mac OS X 10.7.3
- Permisos de administrador al momento de la instalación

7.2.2.3 Linux

Las siguientes son algunas versiones de Linux compatibles:

- Oracle Linux 5.5 en adelante
- Oracle Linux 6.X (32 y 64 bits)
- Red Hat Enterprise Linux 5.5 en adelante (32 y 64 bits)
- Ubuntu Linux 10.04 en adelante

7.3 Plataformas objetivo finales

Tomando como consideración adicional que el desarrollo de la aplicación está dado por un desarrollador independiente, la selección final de plataformas objetivo está a su vez limitada a la posibilidad de adquisición de las licencias de las plataformas mencionadas anteriormente.

Con base en lo anterior se puede declarar que el desarrollo para plataformas de escritorio estará limitado al sistema operativo Windows debido a que tiene mayor distribución y es de más fácil adquisición en términos económicos. De manera similar en el caso de las plataformas móviles, Android es la opción más accesible debido a que su licencia requiere un pago único en contraste con la licencia de iOS, sin mencionar que para desarrollar en iOS es forzosa la adquisición de un Mac, lo cual eleva en gran medida el costo de desarrollo.

En conclusión, se seleccionaron como plataformas objetivo final a Windows en su versión Windows 7 o menor y Android en su versión Jelly Bean o menor.

Capítulo 8: Elaboración de la interfaz

A todo aquél elemento virtual interactivo dentro de un programa que permite la ejecución de tareas o comandos se le conoce como interfaz gráfica de usuario o GUI por las siglas en inglés de Graphic User Interface. Éstas tienen como objetivo brindar al usuario de una aplicación una manera intuitiva de interactuar con las características del programa y facilitar el uso del mismo.
Las GUIs se caracterizan por ser entidades simples que expresan su funcionalidad implícitamente mediante los elementos gráficos que la conforman. Pueden llegar variar visualmente de acuerdo a diferentes aspectos de la aplicación que van desde los objetivos de ésta, la plataforma en la que se despliega y los dispositivos disponibles para interacción, sin embargo hay ciertas características básicas que toda GUI debe cubrir independientemente de los aspectos del programa :

- Facilidad de uso y aprendizaje.
  La interfaz de usuario debe contener símbolos que el usuario pueda asociar fácilmente con la función que se realiza. Esto lo ayudará a aprender a utilizar el software de manera intuitiva.

- Ingreso de la información estrictamente necesaria.
  No se debe saturar al usuario con solicitudes de información que puedan hacer tedioso el uso de la interfaz, ésta se creó para facilitar las cosas no para complicarlas.

- Proporcionar únicamente la información solicitada.
  Arrojar un resultado lo más cercano posible a lo que espera el usuario al utilizar una herramienta, no mas de lo que se puede intuir de la herramienta seleccionada.

- Retroalimentación para el usuario
  El usuario debe saber que el programa está respondiendo a las acciones que éste ejecuta por lo que se debe agregar a la GUI alguna característica que indique al usuario que su comando ha sido recibido, por ejemplo el sombreado de el botón presionado o un mensaje de error en caso de no poder realizar alguna acción.

8.1 Diseño visual de la interfaz

Recordemos del capítulo 6 de esta tesis que dado el hecho de que la aplicación está orientada a ser multiplataforma las interfaces de control seleccionadas son el mouse y

la pantalla táctil ya que comparten gestos similares. Debido a esto, el diseño visual de la interfaz se centrará en elementos con características orientadas a los dispositivos móviles, es decir se omitirán elementos complejos como barras de menús y se optará por botones sencillos con elementos intuitivos para el usuario, todo esto con el fin de manejar el programa con los mismos gestos de entrada y brindar una experiencia de control similar independientemente de la plataforma.

La distribución de los botones en pantalla se realizará de acuerdo a la forma en que un usuario puede tomar un dispositivo móvil de tamaño estándar, de pantalla de 4 pulgadas aproximadamente, ya sea con una o dos manos y al alcance que puede tener con los dedos de la mano.

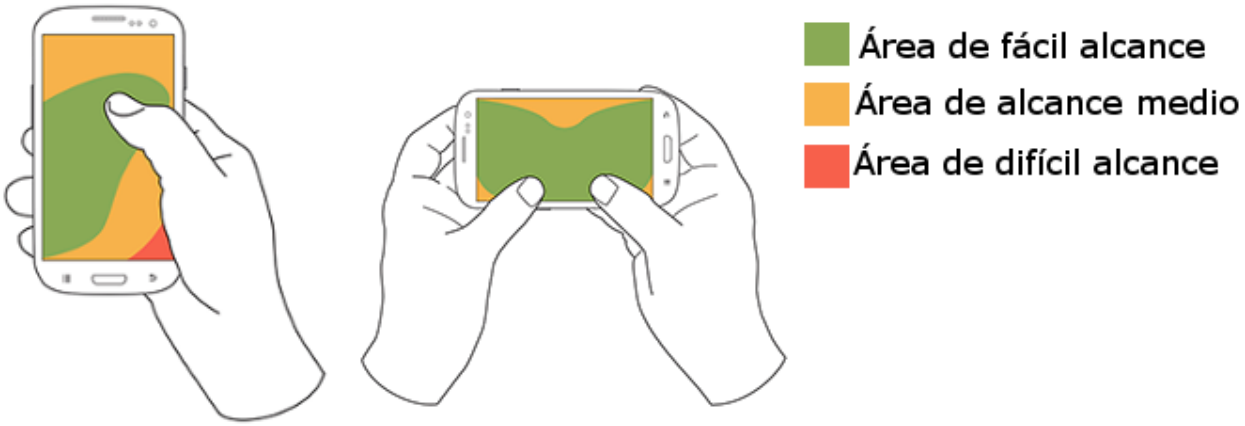


Figura 32: Alcance de los dedos según la forma de tomar el dispositivo.

Imágenes de Uxmatters (www.uxmatters.com)

De la imagen se puede apreciar que el área ideal es la parte superior de la pantalla, tomando como consideración especial que los botones con funcionalidad recurrente se deben localizar más al alcance que los menos frecuentes.

Visto en la aplicación, la distribución de la interfaz de usuario se realiza de la siguiente manera:

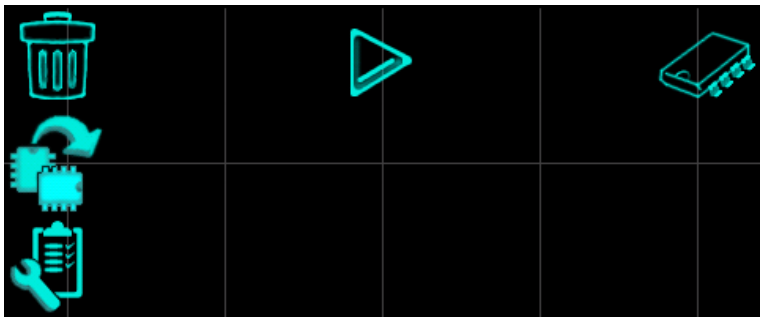


Figura 33: Distribución de interfaz de usuario

En el armado de un circuito, el agregado, edición y supresión de piezas son las funciones más recurrentes, por ello los botones que realizan dichas funciones se encuentran en las esquinas y costados de la pantalla, zonas claramente accesibles, mientras que funciones menos recurrentes como el inicio o detención de la simulación se encuentra al centro.

A continuación se muestran dos capturas de pantalla con la forma de la GUI de la aplicación final con los elementos básicos con los que debe cumplir.

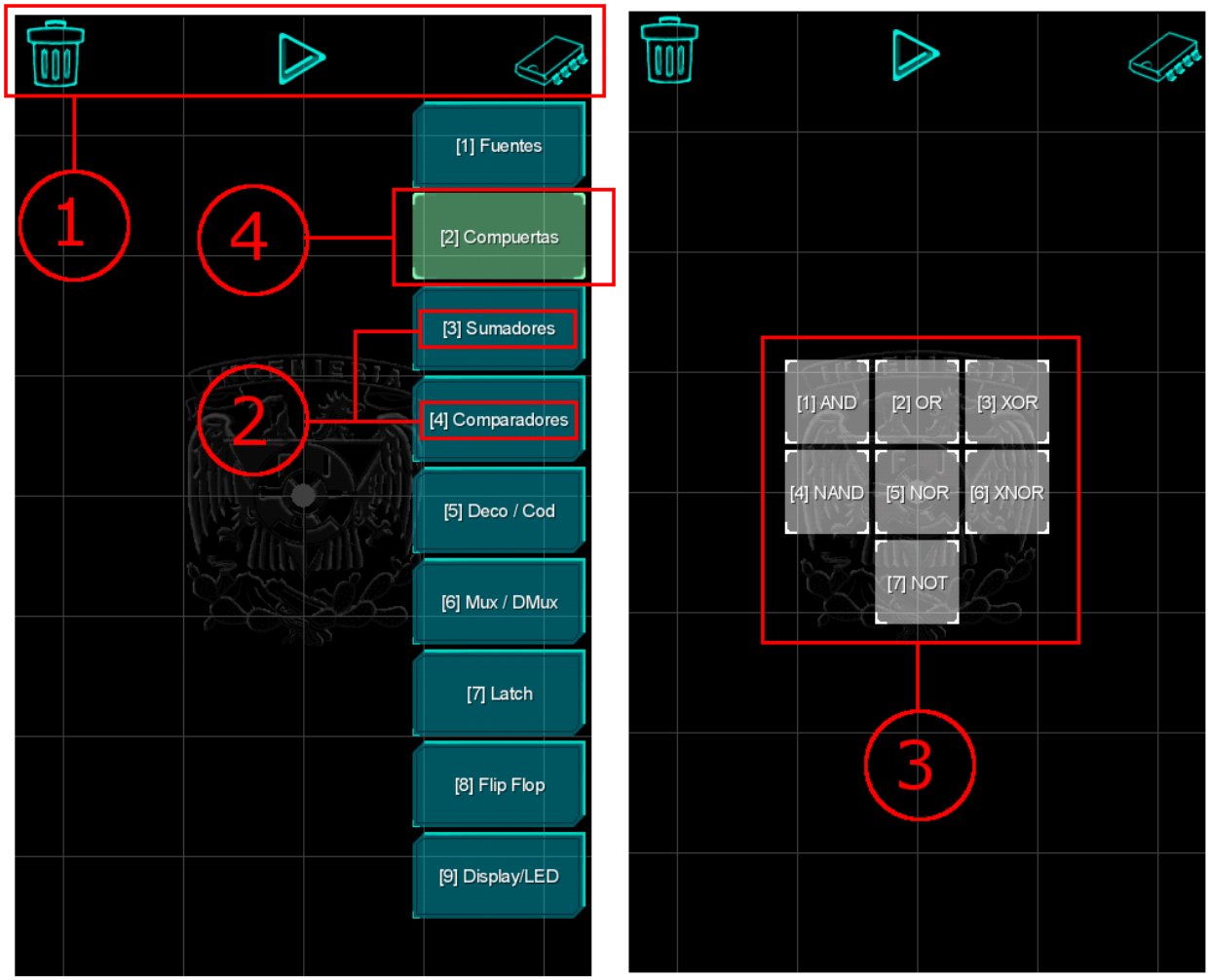


Figura 34: Interfaz de usuario para la aplicación final

1. Facilidad de uso y aprendizaje.

   Los gráficos correspondientes a cada uno de los botones ilustran su funcionamiento. Se pueden apreciar los botones de eliminado, play y nuevo componente. El mismo principio se utiliza en los botones rotar y configurar.

2. Ingreso de la información estrictamente necesaria.

   La interfaz muestra solamente la información necesaria para describir su función. En el caso de las herramientas se utilizan dibujos representativos, sin embargo al no haber un gráfico específico para describir una familia de componentes se utiliza una descripción textual.

3. Proporcionar únicamente la información solicitada.

   El funcionamiento de la interfaz despliega únicamente la información que esta describe. Por ejemplo el menú “compuertas” despliega un sub menú con componentes pertenecientes únicamente a la familia de las compuertas lógicas.

4. Retroalimentación para el usuario

   Se utiliza un sombreado distinto en la interfaz al momento de interactuar con ésta de tal manera que el usuario reconozca que su interacción ha sido registrada.

   Además se incluyen mensajes de aviso, advertencia y error como el que se muestra a continuación.

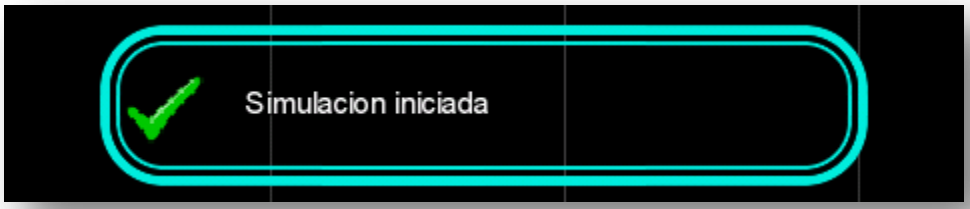
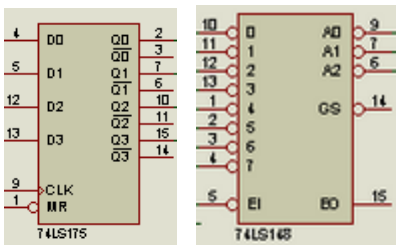


Figura 35: Ejemplo de aviso

8.2 Simbología utilizada en sistemas digitales

Todas las piezas utilizadas en los circuitos digitales cuentan con simbología representativa de las mismas, algunas de ellas con formas particulares como en el caso de las compuertas lógicas, sin embargo al existir una infinidad de piezas, la simbología tiende a una forma general derivada de la complejidad del componente y basada en la descripción de sus pines[10]. Esta forma se define por un rectángulo representativo del cuerpo del circuito integrado y de sus pines correspondientes acomodados de un lado u otro de este dependiendo de si se trata de un pin de entrada de datos o de salida de datos. A continuación se muestran ejemplos de ésta simbología utilizados en Proteus, ISIS.

[10] Patitas de un circuito integrado mediante las cuales se realizan conexiones entre componentes



Representación rectangular de componentes en Proteus, ISIS
entradas del lado derecho y salidas del lado izquierdo

Cabe mencionar que dicha simbología rectangular se asemeja mucho a la forma física de un circuito integrado real, sin embargo ésta no es la misma y no se debe considerar como la distribución final de pines en el circuito real al momento de alambrarlo[11]. La distribución de pines en un circuito físico se puede encontrar en las hojas de datos del componente en cuestión, éstas son mejor conocidas como *Data sheets* y contienen información del funcionamiento del componente.

Basado en las características descritas de la simbología utilizada en simuladores existentes, el diseño para los símbolos de la aplicación se define como se muestra en la siguiente imagen, respetando la simbología característica de las piezas básicas y utilizando la forma rectangular general para componentes más complejos.

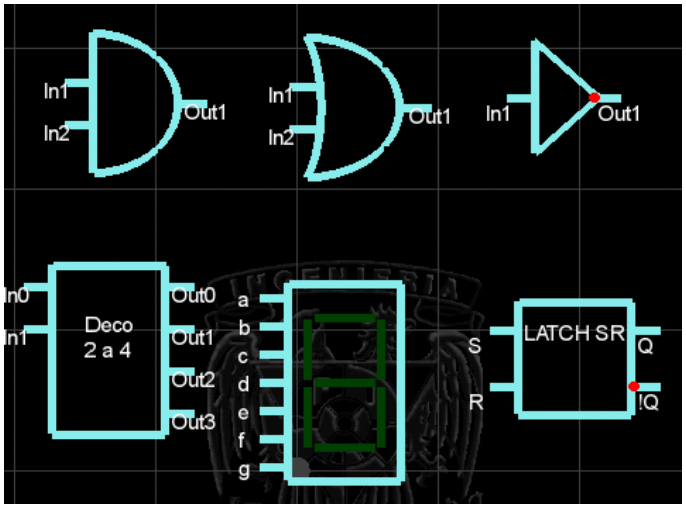


Figura 36: Imagen con diferentes piezas disponibles en la aplicación.

A la izquierda se encuentran los pines de entrada y a la derecha los pines de salida.

8.3 Estudiando interfaces existentes

Las interfaces de usuario de los programas de simulación existentes ofrecen herramientas que brindan funcionalidad extra para realizar diferentes tareas o para

[11] Alambrar: Construir el circuito con todas sus conexiones en una tabla de prototipado.

visualizar datos en formas variadas, sin embargo, la mayoría de éstas coincide en una forma básica común de control para llevar a cabo la construcción de un circuito: selección del componente, posicionamiento libre en un lienzo virtual y alambrado de las conexiones. Este tipo de comportamiento básico le ofrece al usuario una sensación natural de construcción muy cercana a la que se experimenta al dibujar el circuito en papel, es por ello que se toma como forma interactiva básica en una aplicación y por lo que se utilizará de igual manera para la aplicación de esta tesis.

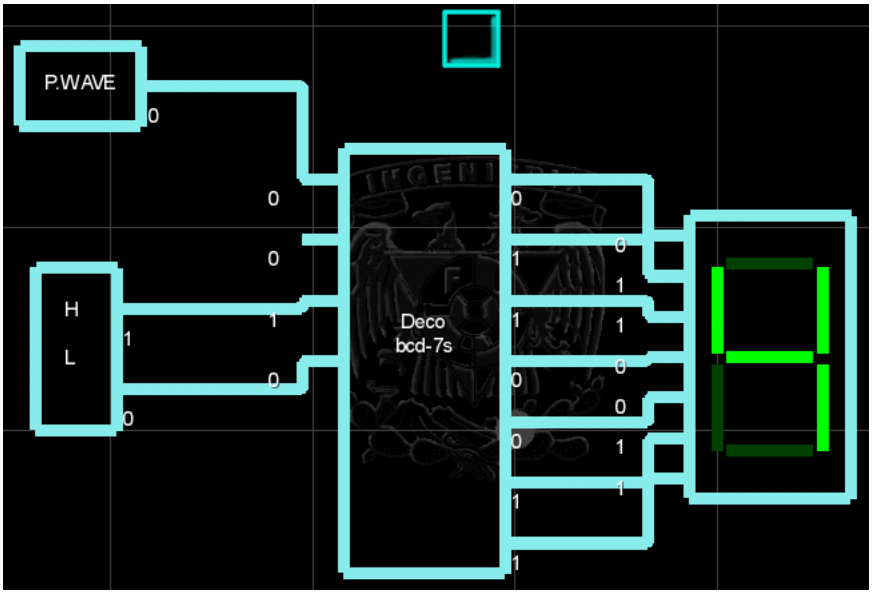


Figura 37: Ejemplo de circuito

Otras herramientas de edición en un programa dependen en gran medida de la plataforma en la que se ejecuten, debido a características como la distribución en pantalla y las capacidades del dispositivo.

8.3.1 Interfaces de escritorio

En las GUIs para plataformas de escritorio se cuenta con mayor libertad de integración gracias al hecho de que las pantallas de los dispositivos en estas plataformas suelen ser más grandes y por lo tanto la visión de los elementos relevantes para el usuario no se ve afectada gravemente, inclusive las capacidades del dispositivo son mayores y en consecuencia se puede agregar una gran cantidad de herramientas extra ejecutables simultáneamente sin que el rendimiento se vea afectado.

Figura 38: GUI de escritorio

Al observar la imagen de la GUI de Proteus, ISIS, se puede apreciar la gran cantidad de botones disponibles para el uso de distintas herramientas, todos los cuales tienen una distribución tal que el espacio utilizado sigue siendo menor al espacio de mayor relevancia para el usuario, el espacio de edición.
En la versión de escritorio de la aplicación de esta tesis una interfaz compleja no es necesaria debido a que las herramientas disponibles en ella no son tantas como para construir dicho tipo de interfaz.

8.3.2 Interfaces móviles

Las interfaces gráficas en dispositivos móviles suelen requerir de mayor cuidado al realizar la distribución de sus componentes ya que estos están diseñados precisamente para ser portables y esta simple característica conlleva dos factores físicos determinantes en el desarrollo de una aplicación: el tamaño de pantalla y la capacidad de procesamiento.
Al tener los dispositivos móviles un tamaño portable la pantalla de los mismos se ve reducida, dejando un menor porcentaje espacial dedicado a la interfaz de usuario, limitando al diseñador de la interfaz a utilizar solo aquellos elementos estrictamente necesarios.
En cuanto a capacidades de procesamiento, los componentes en un dispositivo móvil son más pequeños para no comprometer su portabilidad, sin embargo esto limita a dichos componentes a tener capacidades de procesamiento menores que las encontradas en dispositivos no portables. Este aspecto influye en el diseño de una aplicación móvil en cuanto a que el programador debe tomar en cuenta que no puede realizar cargas de trabajo al procesador de la misma forma en que lo haría en una plataforma de escritorio, de lo contrario el desempeño de la aplicación se verá comprometido.

A continuación se muestra un ejemplo de cómo se limita una interfaz de acuerdo a la plataforma utilizando como ejemplo la aplicación web de Google Docs en su versión para computadoras de escritorio y para dispositivos móviles.

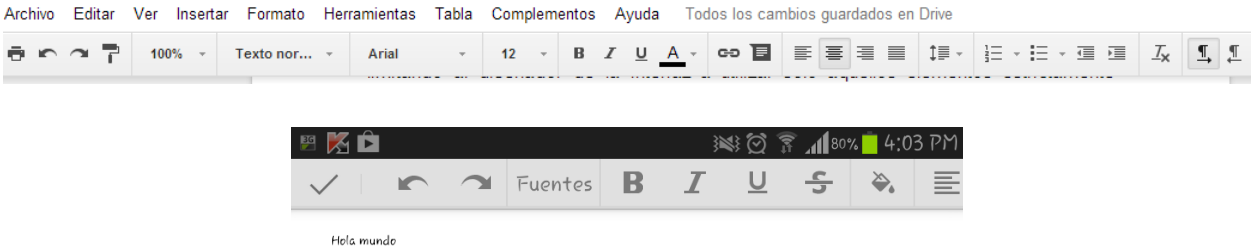


Figura 39: GUI de Google Docs versión de escritorio (arriba) y versión móvil (abajo).

Para la versión móvil de la aplicación el diseño de la interfaz se mantendrá como se

describe al inicio de este capítulo en el sub tema **Diseño visual de la interfaz**.

8.4 Controles de la interfaz

Tomando como referencia a las aplicaciones Proteus (ISIS) y Every Circuit, se puede apreciar que para la manipulación de las piezas disponibles el control de la interfaz permite libertades de edición como el posicionamiento de piezas, rotación y elaboración de conexiones. Todas estas características se agregaron a la aplicación final como controles de comportamiento similar basados en interfaces existentes con el fin de que otros usuarios con experiencia en el uso de este tipo de herramientas se sientan cómodos y familiarizados con la aplicación.

En la siguiente imagen se aprecia la similitud en los elementos de control de la aplicación final con respecto a la aplicación móvil Every Circuit.

Figura 40: Similitudes entre interfaces

8.5 Controles de interfaces existentes

Muchos de los simuladores de circuitos, ya sea analógicos o digitales facilitan al usuario final una forma de control que varía dependiendo de la plataforma en la que se ejecuten, éstas se describirán en los subtemas siguientes de éste mismo capítulo dependiendo de si se trata de interfaces de escritorio o móviles, sin embargo hay una característica en la que todos los simuladores coinciden y que vale la pena describir antes de pasar a analizarlas por separado: el lienzo de edición.

El lienzo de edición es el área del simulador en la cual se pueden agregar piezas, establecer conexiones y editar características específicas de las mismas como por ejemplo su rotación o su funcionamiento. El comportamiento del lienzo de edición es muy similar independientemente de la plataforma en la que se esté ejecutando la

aplicación, éste facilita controles como los siguientes:

- Posicionamiento y reubicación de componentes
- Supresión de componentes
- Elaboración de conexiones
- Presentación de datos

Todas estas características se podrían considerar como mínimas básicas para un lienzo de edición en un simulador de circuitos debido a que con ellas el usuario tiene el control necesario sobre la aplicación para la construcción de un modelo de circuito. Es por esto que dichas características se agregaron a la forma de control de la aplicación de esta tesis. Las siguientes imágenes ilustran algunas de las características mencionadas anteriormente aplicadas al lienzo de edición de la aplicación final.

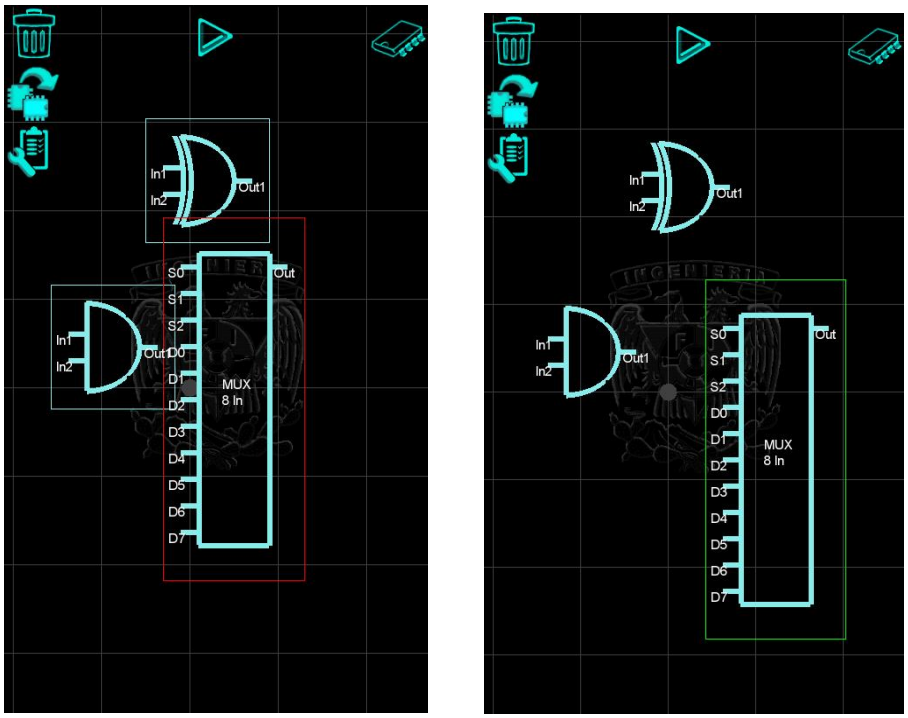


Figura 41: Ejemplo del posicionamiento de componentes en el lienzo de edición.

El traslape de componentes se indica mediante elementos gráficos.

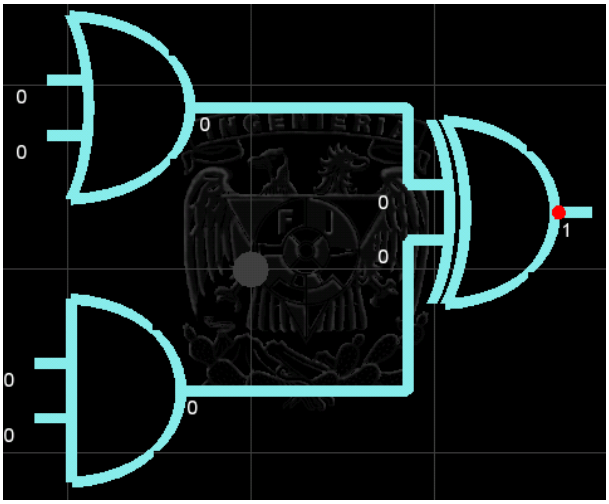


Figura 42: Forma de presentación de información.

Ceros para representar niveles bajos y unos para representar niveles altos.

8.5.1 Interfaces de escritorio

En una interfaz de escritorio los dispositivos periféricos como el ratón y el teclado estarán presentes en la mayoría de los casos. Tomando en consideración que el usuario tiene dichos dispositivos como principal medio de control en una interfaz de escritorio la aplicación final también integra funciones que permiten el uso de estos.

En cuanto al uso del ratón, la aplicación usará gestos de control como se describe anteriormente en el capítulo 6 de esta tesis, de manera tal que se pueda tener un control total de las herramientas disponibles mediante la interfaz gráfica y a la vez garantizando una experiencia similar al interactuar en plataformas móviles.

Con respecto al teclado, la aplicación usará atajos de teclado para la activación de funciones sin necesidad de interactuar directamente con la GUI. En este caso, la GUI servirá como una referencia en la que se pueden apreciar las teclas que corresponden a cada función.

El uso de las interfaces de mouse y teclado se realiza de manera complementaria, es decir, se pueden realizar acciones mediante el uso del mouse con la GUI y a la vez utilizar el teclado para concretar una acción.

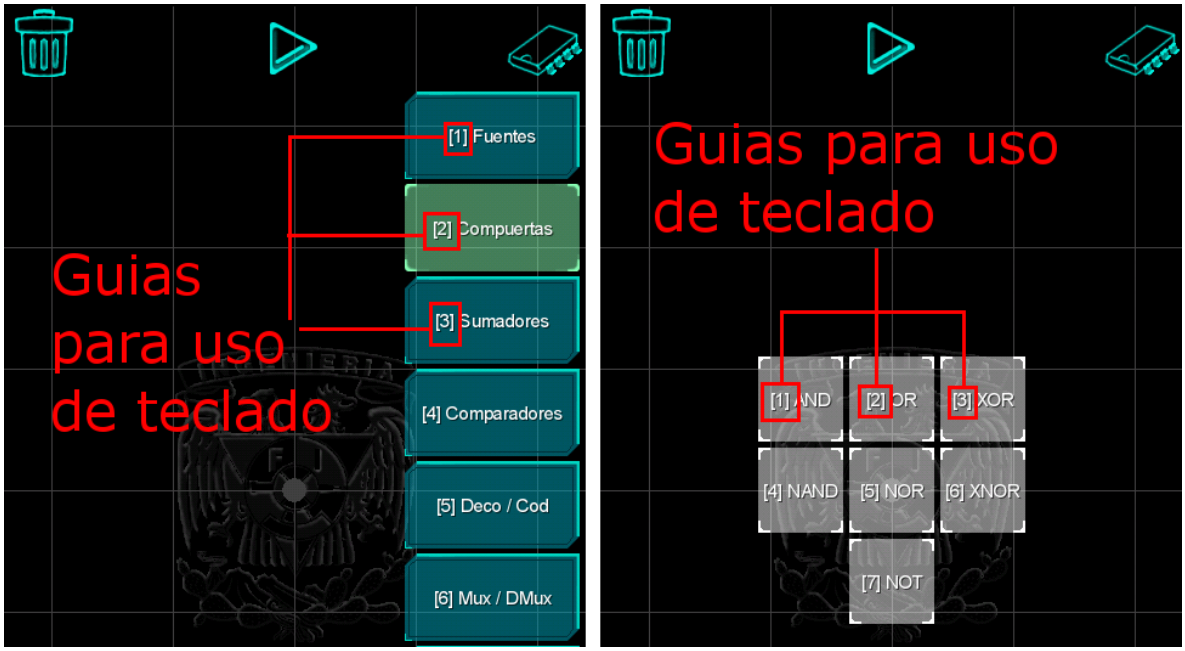
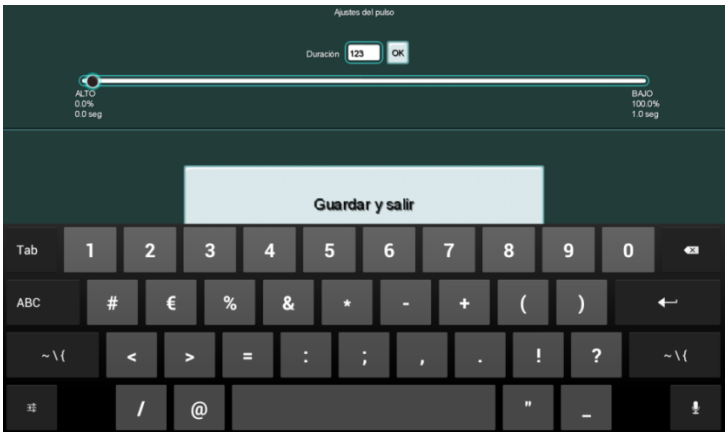


Figura 43: Guías para uso de atajos de teclado

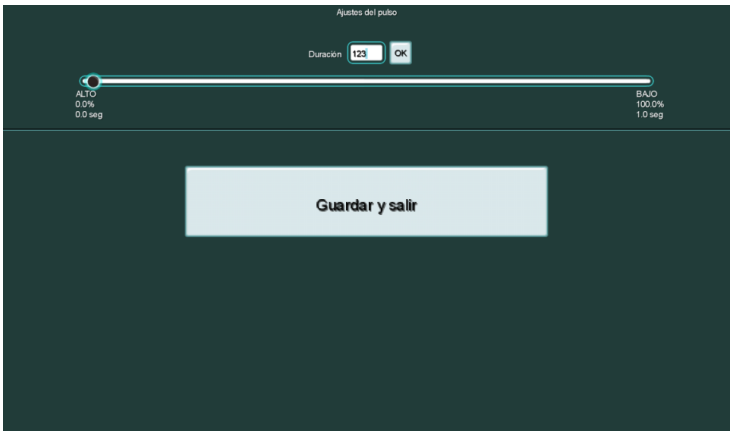
8.5.2 Interfaces móviles

En una interfaz móvil el control de la aplicación depende en gran medida de los gestos táctiles y su interacción con la interfaz gráfica debido a que no todos los dispositivos móviles incluyen periféricos como los de una computadora de escritorio, sin embargo aquellos que cuentan con la capacidad de soportar la conexión de alguno de estos dispositivos se verá beneficiado de la forma de control que ofrecen en una interfaz de escritorio ya que la aplicación está diseñada para reconocer cuando se tiene conectado un dispositivo externo e integrar la funcionalidad de dicho elemento como extra a la forma de control táctil.

En la siguiente imagen se muestra un ejemplo de cómo se realiza la integración de un teclado en un dispositivo móvil y cómo la aplicación responde desplegando un teclado virtual al no tener nada conectado y omitiendo el despliegue del teclado virtual al tener conectado el teclado físico.



Sin un teclado físico conectado surge el teclado virtual.
Se puede ver la inserción de texto en el espacio “Duración”



Con un teclado físico conectado no surge el teclado virtual.
Se puede ver la inserción de texto en el espacio “Duración”

Capítulo 9: Estructura del programa

LibGDX permite el uso de un sistema de pantallas de aplicación dentro de las cuales se pueden definir grupos jerárquicos de objetos, igual que en un árbol de escena. Se decidió hacer uso de estas características ya que facilitan la detección y en su caso la contención de elementos específicos sin que se vea comprometida la integridad total de la aplicación. La estructura final del programa se ilustra en el siguiente diagrama jerárquico.

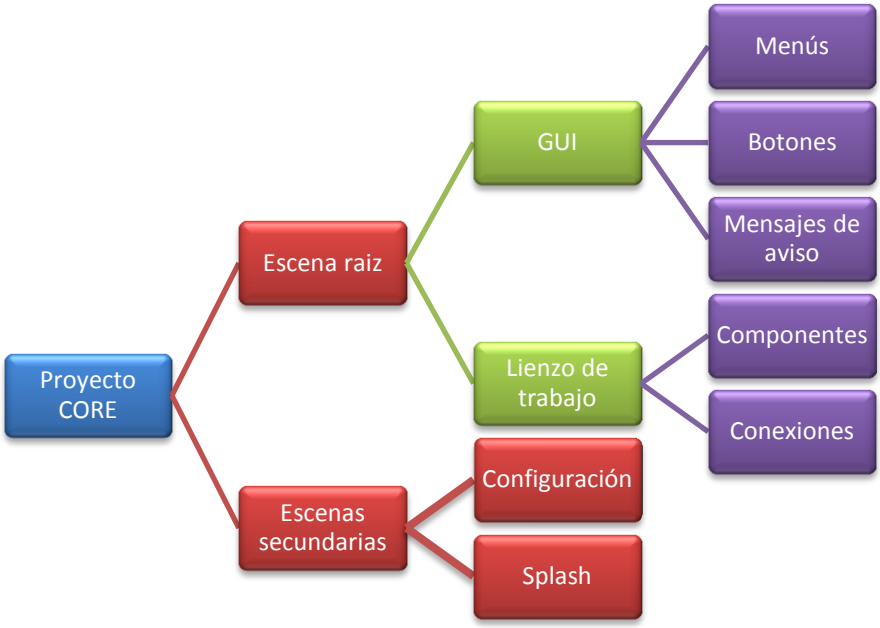


Figura 44: Árbol jerárquico de la aplicación

En él se puede observar un nodo raíz llamado “Proyecto CORE”, éste es el proyecto principal, característico del framework LibGDX, en donde se define la lógica del programa. Los siguientes dos nodos “Escena Raiz” y “Escenas secundarias” son las pantallas principales que utiliza la aplicación y que están contenidas en la lógica principal. El nodo “Escenas secundarias” contiene las pantallas de configuración y presentación (o *splash*) las cuales son llamadas a desplegarse en diferentes momentos durante la ejecución del programa. El nodo “Escena raíz” contiene grupos de objetos de funcionalidad afín, por ejemplo el grupo “GUI” contiene todo lo relacionado con la interfaz de usuario, mientras que el grupo “Lienzo de trabajo” contiene objetos característicos del área de edición y simulación.

Al ser la aplicación de esta tesis un desarrollo sin precedentes, se seleccionó la metodología de desarrollo de software de *Modelado en Cascada*. El Modelado en Cascada consiste en la imposición de una serie de pasos que deben seguirse en orden secuencial, en donde cada paso es una etapa del proyecto que debe realizarse completamente antes de poder seguir con las demás y que debe someterse a pruebas para su verificación y en su caso mantenimiento. Esta metodología facilita el proceso

de construcción de elementos básicos primordiales de la aplicación y que dan pie a características más complejas.
Por ejemplo, para realizar un acercamiento de cámara (zoom in) primero se debe detectar el input[12] generado por el usuario, posteriormente identificar el gesto implicado en el input (toque, arrastre, scroll, etc) y finalmente realizar la acción asociada al zoom o en su defecto a la acción correspondiente.

Figura 45: Ejemplo de acciones a seguir para la detección de entradas

En cuanto a las pruebas realizadas para la verificación de las etapas de desarrollo se llevaron a cabo pruebas de usuario en dos categorías: Experiencia por plataforma y Funcionalidad. En ambas categorías los usuarios eran tanto personas con conocimientos técnicos con experiencia en el manejo de dispositivos como personas sin conocimientos técnicos familiarizadas con el uso de un solo tipo de dispositivo.
La primera categoría somete a la aplicación a usuarios de plataformas específicas para revisar la forma en que éstos respondían ante el la forma de control de la aplicación. La segunda categoría evalúa la eficiencia y comodidad de uso de la función en cuestión.

9.1 El Paradigma orientado a objetos

Existen diferentes paradigmas de programación utilizados en la construcción de programas computacionales, en el caso de la aplicación de ésta tesis el paradigma a utilizar es el llamado Paradigma Orientado a Objetos, abreviado POO, debido a que LibGDX, el framework utilizado en la construcción de la aplicación, está basado en Java, lenguaje de programación con paradigma orientado a objetos, pero además debido a que el POO cuenta con cuatro características útiles para la construcción de la aplicación conocidas como abstracción, herencia, encapsulamiento y polimorfismo.

La primera característica es la **abstracción**. En un simulador de circuitos los componentes pueden ser requeridos en más de una ocasión y el POO permite al programador crear un archivo de código llamado *clase* en el cual se abstraen las características de un componente para posteriormente crear instancias de dicha clase que pueden existir en el programa como elementos independientes.

[12] Datos de entrada generados por la interacción del usuario con el o los dispositivos que le permiten manipular la aplicación.

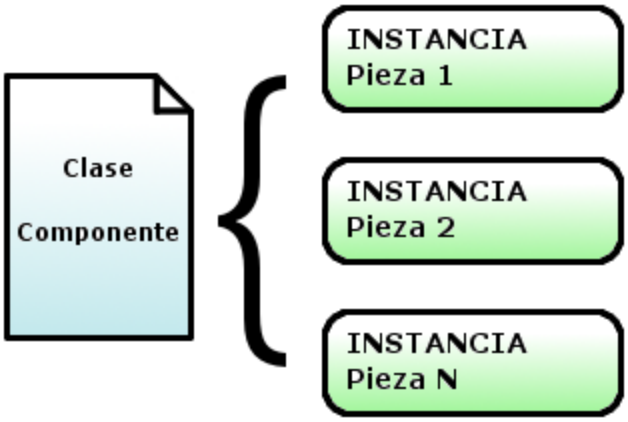


Figura 46: Clase con instancias

La segunda de las ventajas presentes en el POO es conocida como **herencia**. La herencia es una propiedad que permite traspasar las características de una clase base a otras clases derivadas de ésta. A la clase de la que se derivan otras clases se le conoce como clase padre y a las derivadas se les conoce como hijas. La herencia permite reutilizar código anteriormente descrito y evita su redefinición. Por ejemplo, imaginemos dos componentes electrónicos: una compuerta lógica y un decodificador; a pesar de ser la segunda una pieza más compleja que la primera ambas tienen una característica en común: los pines. Si creamos una clase que tenga como característica un componente con pines, podremos definir subclases derivadas de ésta que hereden la característica de los pines en las que a su vez se defina un comportamiento único de la pieza.

Las dos características restantes, **encapsulamiento** y **polimorfismo**, son propiedades del POO cuya aplicación es mejor apreciable a nivel de código. La primera se refiere a la restricción de los datos de una clase para evitar su uso no autorizado por elementos externos y la segunda se refiere a la redefinición de estructuras de código para adaptarlas a fines diversos.

Estas características del POO se pueden aplicar a muchos otros aspectos del desarrollo de la aplicación que van desde la funcionalidad de cada pieza hasta el diseño gráfico de los componentes y son las que lo vuelven propicio para uso en una aplicación de éste tipo.

9.2 Escenas en la aplicación

El framework LibGDX ofrece el manejo de escenas para el despliegue de información, una escena es una clase que contiene elementos gráficos de todo aquello que se quiere mostrar al usuario en un determinado momento. Las escenas son útiles especialmente en interfaces móviles ya que permiten el uso de la totalidad de la

pantalla del dispositivo para mostrar GUIs de una manera mucho más eficiente.

Para la aplicación final el uso de escenas se implementa para pantallas de configuración como se muestra en la imagen siguiente.

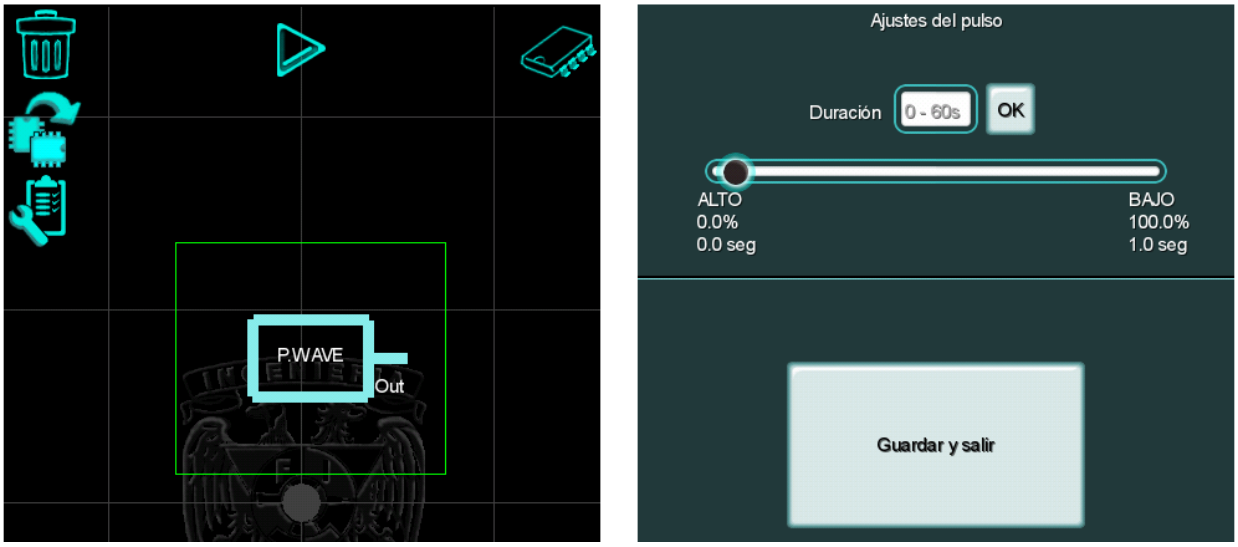


Figura 47: A la izquierda pantalla de edición con una pieza seleccionada.

A la derecha pantalla de edición de la pieza seleccionada.

9.3 Detección de entradas

Como se mencionó anteriormente, el control de la aplicación se basará en los gestos comunes realizables por el mouse y la pantalla táctil, sin embargo también se incluirán otras formas de control dependiendo de algunos gestos particulares característicos de cada plataforma a los que el usuario puede estar mejor acostumbrado, por ejemplo el gesto pinch en dispositivos móviles y el scroll de mouse en plataformas de escritorio para controlar el zoom de la cámara.

A continuación se describen cada una de las funciones de control disponibles en la aplicación y la forma de realizarlas de acuerdo a la plataforma.

Función	Móvil	Escritorio
Selección	Tap	Clic
Arrastre	Touch Drag	Mouse Drag

Zoom	Pinch	Mouse scroll
Atajos de teclado	Teclado	Teclado

Tabla 3: Funciones de control

Existe la posibilidad de que a un dispositivo móvil se le integren periféricos característicos de computadoras de escritorio como un mouse o un teclado, en estos casos la respuesta de los controles será la misma definida en la tabla anterior.

9.3.1 Versión de escritorio

Como se describe en el capítulo anterior, la aplicación cuenta con atajos de teclado que facilitan el manejo de ésta en sistemas operativos de escritorio, cada uno de estos atajos está definido para realizar una función en la aplicación de acuerdo a las guías que brinda la interfaz gráfica.

La forma de mapear dichos atajos con su correspondiente función se realiza de manera dinámica, es decir depende del estado en que se encuentre la aplicación. Al hacer el mapeo de esta manera se pueden reutilizar teclas, garantizando que un número mayor de funciones contenga un atajo y facilitando el aprendizaje al usuario final al no tener que memorizar una tecla en específico por cada función que se quiera realizar.

Por ejemplo, de la siguiente imagen se puede observar que las guías de los atajos de teclado indican que la tecla del número 2 se utiliza tanto en el menú principal (imagen de la izquierda) como en el sub menú de selección de componente (imagen de la derecha). Esto significa que dependiendo de si la aplicación se encuentra en un estado de **despliegue de menú principal** o de **selección de componente** la tecla con el número 2 tendrá una función diferente.

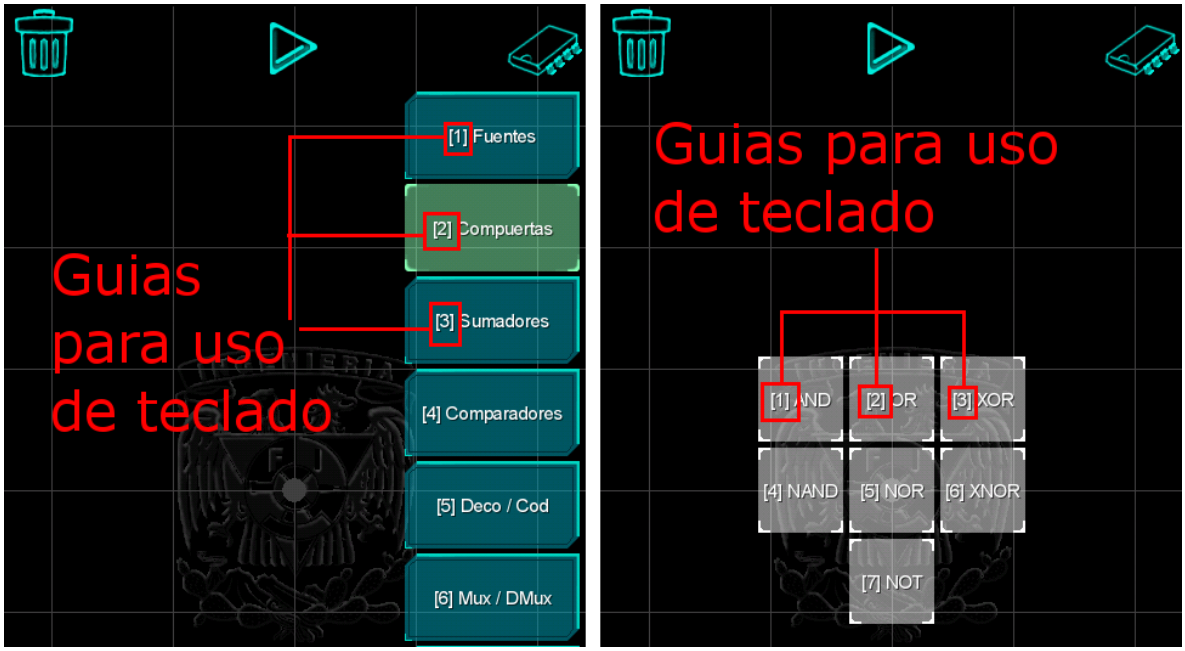


Figura 48: Guías para atajos de teclado

9.3.2 Versión móvil

La forma de detectar entradas de la aplicación en la versión móvil, considerando que no se tengan periféricos conectados al dispositivo móvil, está limitada a los eventos touch que se realicen en la pantalla táctil del dispositivo. Esto conlleva una detección de eventos por niveles, es decir, se debe verificar cada evento táctil al momento de ejecutarse para determinar en qué elemento de la aplicación recae la acción. Por ejemplo, si una pieza se encuentra ubicada en pantalla de manera tal que su posición se traslapa con el de un botón de la interfaz gráfica, la interfaz tenga prioridad de ejecución al haber un evento touch en el área de traslape.

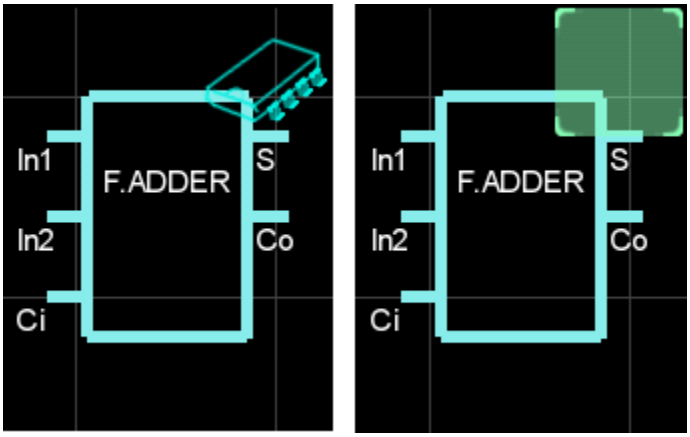


Figura 49: Detección de un evento touch en el área de traslape y reaccionando en la GUI

Algunos de estos eventos táctiles están sujetos al estado de la aplicación, de manera similar a como se hace con los atajos de teclado. Por ejemplo, al activar un sub menú las piezas que se encuentren sobre el lienzo de edición se sombrearán indicando que toda la acción de eventos táctiles recaerá únicamente sobre el menú en cuestión al estar el programa en un estado de selección de componente.

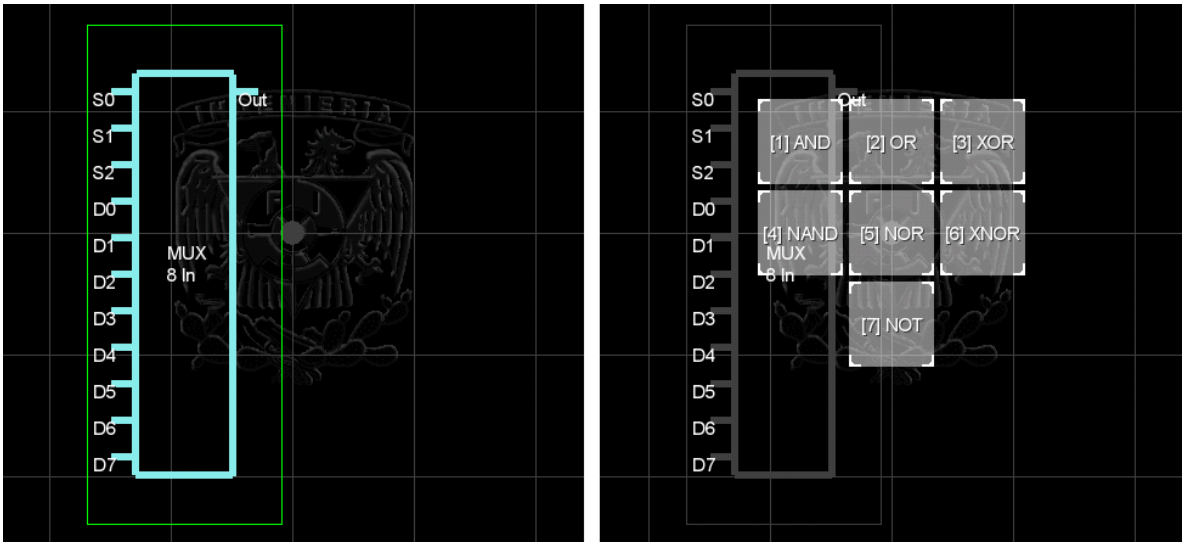


Figura 50: Coloreado de la pieza en gris para indicar que pasa a segundo plano y las acciones recaen en la interfaz gráfica.

9.4 Estructura de los componentes

En la aplicación cada pieza disponible está definida como una instancia de una clase específica que define las características particulares de la misma, sin embargo dicha clase comparte componentes heredados de otras clases que le brindan características comunes a todas las piezas. Esto se logra mediante el uso de herencia

9.4.1 Uso de herencia

El siguiente diagrama muestra a grandes rasgos cómo están estructuradas algunas piezas de la aplicación y la manera en que se relacionan mediante herencia.

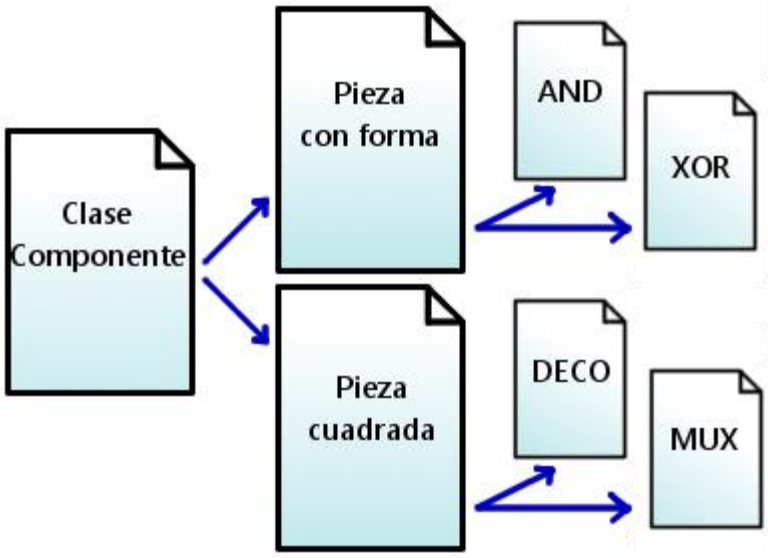


Figura 51: Ejemplo de herencia en clases

De la imagen se puede observar que hay una clase padre llamada “Componente” de la cual derivan todas las demás. En esta primera clase se definen variables comunes a todas las piezas de manera abstracta para posteriormente inicializarlas en sub clases derivadas con los valores específicos según lo requiera el componente. En la primera ramificación surgen dos clases, la primera “Pieza con forma” define a las piezas con formas particulares y la segunda “Pieza cuadrada” define todas aquellas piezas que siguen el estándar de gráfico rectangular discutido en el capítulo 8.

Capítulo 10: Desplegando resultados

La aplicación está estructurada de acuerdo al diagrama de bloques que se muestra a continuación.

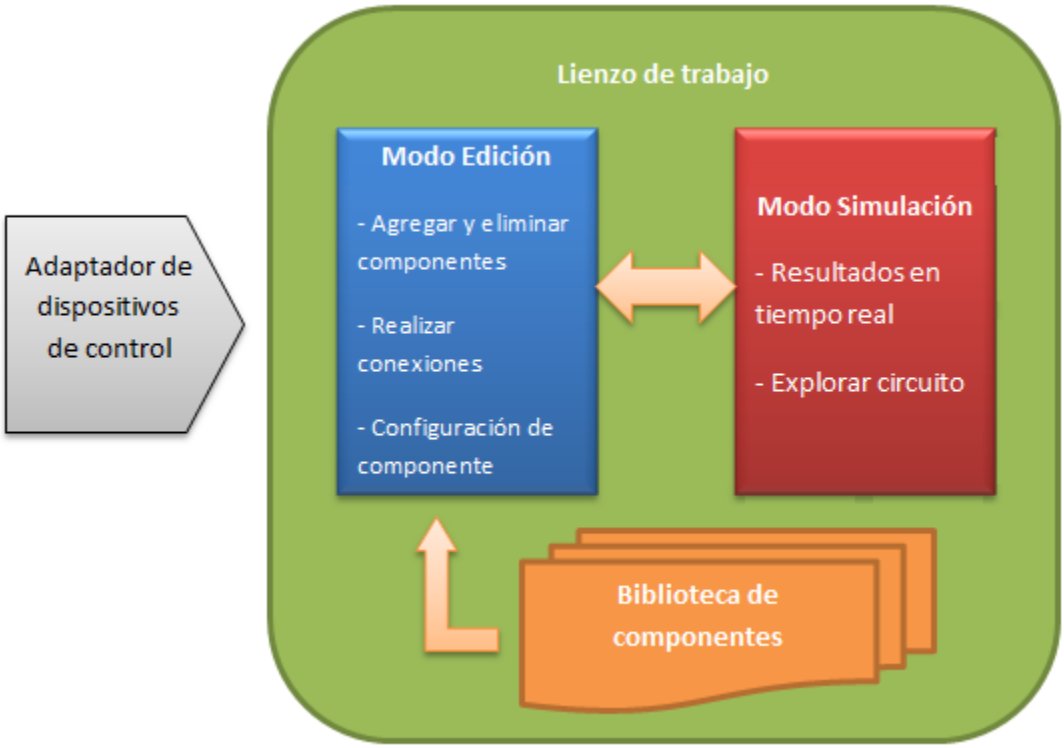


Figura 52: Diagrama de bloques de la aplicación

El primer bloque es un adaptador de dispositivos de control, este recibe las entradas del usuario, además de que permite la compatibilidad en caso de haber diferentes tipos de dispositivos de control disponibles.

El segundo bloque es el lienzo de trabajo, en él se incluyen dos modalidades de uso de entre las que se puede cambiar en cualquier momento: Edición y Simulación.

El Modo Edición permite la construcción de un circuito virtual mediante la adición y/o supresión de componentes y conexiones. Los componentes disponibles están guardados en una biblioteca de componentes y solo pueden ser agregados y configurados en el lienzo de trabajo mientras se esté en modo de edición.

El Modo Simulación se visualizan los resultados realizados por la operación de cada componente y la forma en que estos afectan el comportamiento de otros que a su vez se encuentren conectados. En esta modalidad se bloquea la edición del circuito como una forma de protección para el usuario.

Se puede consultar una lista de las piezas disponibles en la aplicación en el ANEXO 1 de esta tesis.

La aplicación final cuenta con diferentes ventajas destacables con respecto a programas existentes que le permitieron cumplir con los objetivos planteados, todas ellas derivadas del análisis de diversas herramientas de simulación y desarrollo de software descritas a lo largo de ésta tesis y de la retroalimentación obtenida en las pruebas de usuario. Muchas de las ventajas enlistadas a continuación se encuentran en diferentes aspectos de la aplicación que van desde la interfaz de usuario hasta la funcionalidad misma y son el producto final de una serie de cambios generados durante la fase de desarrollo, la mayoría de los cuales se basaron en la recreación de la experiencia de uso de programas de índole similar.

Interfaz instructiva

Comparado con aplicaciones en donde un manual de uso es indispensable para el manejo de la misma, la aplicación de esta tesis cuenta con una interfaz instructiva en la que sus elementos guían al usuario mediante la activación de otros elementos secundarios de manera progresiva, de esta manera la progresión en el uso de la interfaz guía al usuario en la forma de controlarla.

Interfaz intuitiva

Para usuarios que han trabajado antes con simuladores de circuitos la aplicación de esta tesis ofrece una interfaz de edición muy similar a la que presentan la mayoría de los simuladores, esta es la de un lienzo de edición plano con la simbología característica de los circuitos.

Además de la simbología de circuitos, la simbología de la interfaz está diseñada de manera tal que el usuario intuya la funcionalidad de la misma.

Presentación de resultados

La aplicación de esta tesis está orientada al área de los circuitos digitales, por tal razón la forma de representar los resultados se realiza en forma de bits. Esto representa una ventaja con respecto a otras aplicaciones debido a que en algunas de ellas la información que se presenta se encuentra en valores flotantes de voltaje o corriente los cuales deberán ser interpretados posteriormente por el usuario en valores binarios para poder obtener la información de simulación requerida.

Despliegue multiplataforma

La aplicación ofrece la capacidad de ejecutarse en diferentes sistemas operativos tanto de dispositivos móviles como de escritorio ofreciendo una experiencia de uso similar. Esta característica se considera una ventaja con respecto a otras aplicaciones ya existentes debido a que no muchas de éstas cuentan con la capacidad de un despliegue múltiple.

Integración de hardware

La integración de periféricos para el control de la aplicación es una ventaja que está estrechamente ligada al despliegue multiplataforma. Muchas aplicaciones existentes limitan su forma de control a la que ofrezca la plataforma en la que se esté ejecutando, sin embargo para la aplicación de esta tesis la forma de control es adaptable, es decir, se pueden integrar periféricos como teclado y mouse siempre que el dispositivo lo soporte. La aplicación reaccionará de acuerdo a los dispositivos conectados.

A forma de ilustrar el despliegue de la aplicación final se presenta una imagen de la ésta ejecutándose en dos sistemas operativos: Windows y Android, con un circuito de ejemplo.

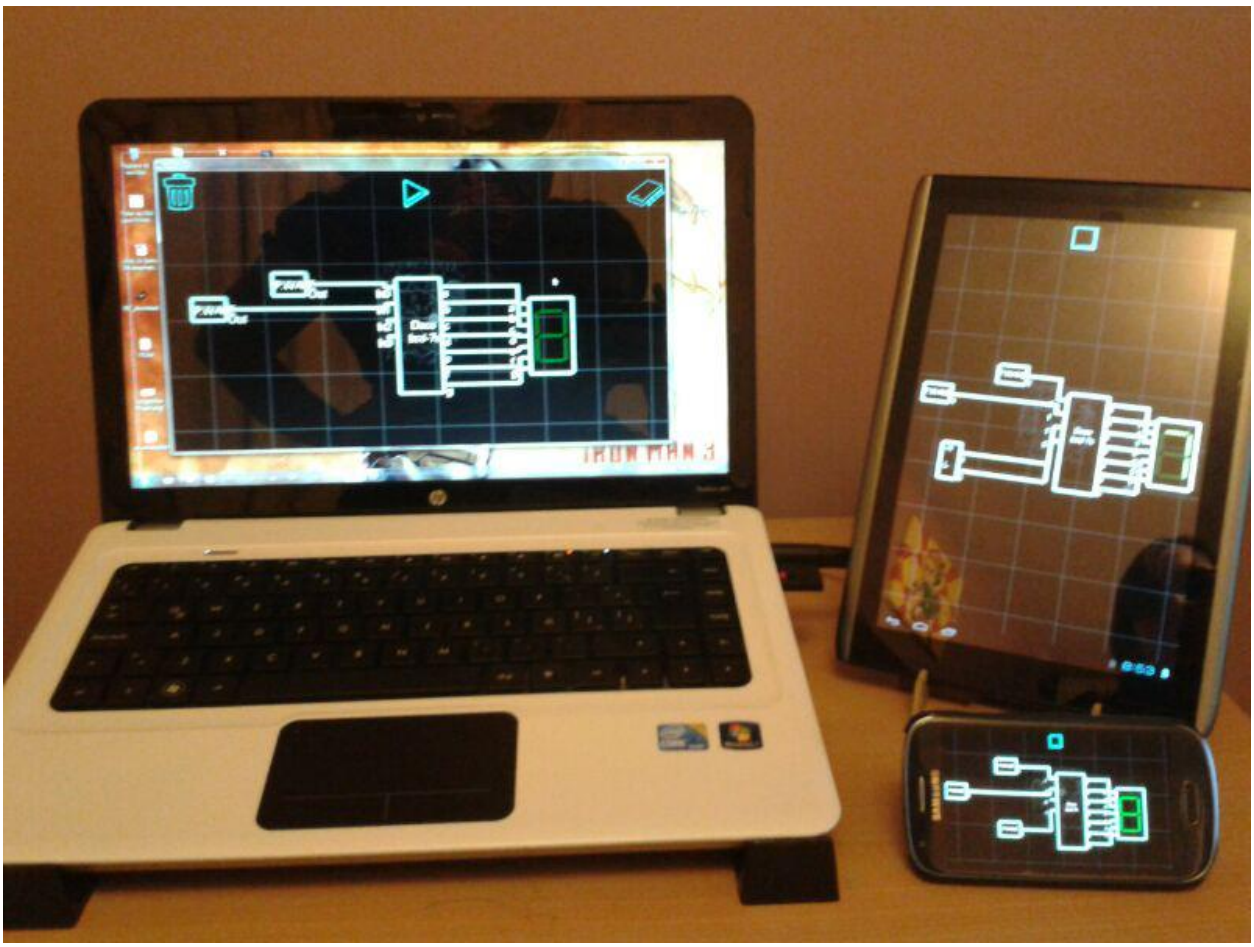


Figura 53: Despliegue en múltiples dispositivos

También se presenta la implementación de algunos ejemplos de circuitos digitales en la aplicación final. Se muestran los esquemas de los ejercicios y una captura de pantalla del simulador con el ejercicio representado.

El primer ejemplo muestra el concepto de la universalidad de la compuerta NAND como elemento lógico universal.

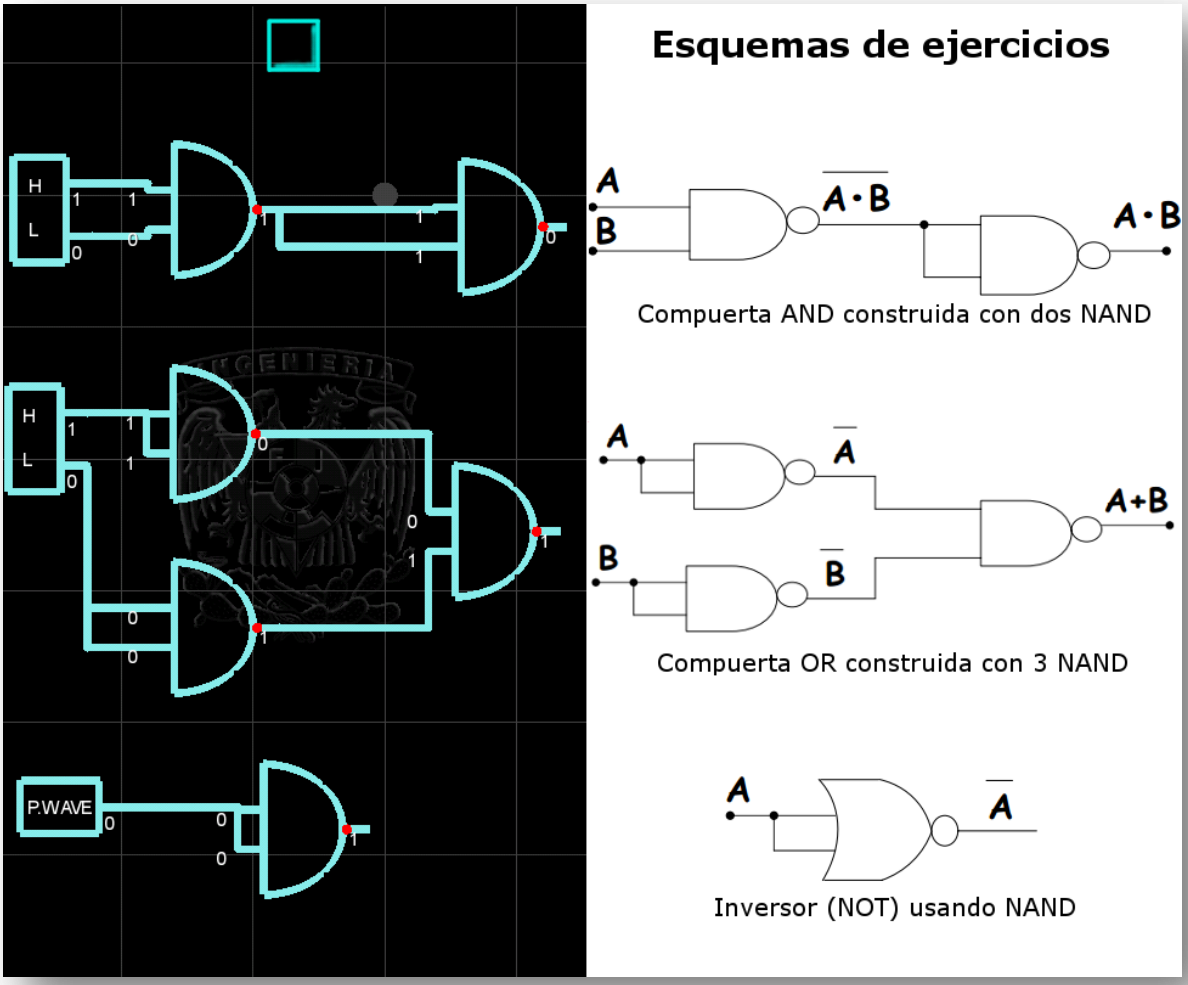


Figura 54: Ejercicio de ejemplo 1

El siguiente ejemplo muestra la construcción de un medio sumador haciendo uso de las compuertas AND, XOR y NAND. Además se muestra uso del componente medio sumador predefinido en la aplicación.

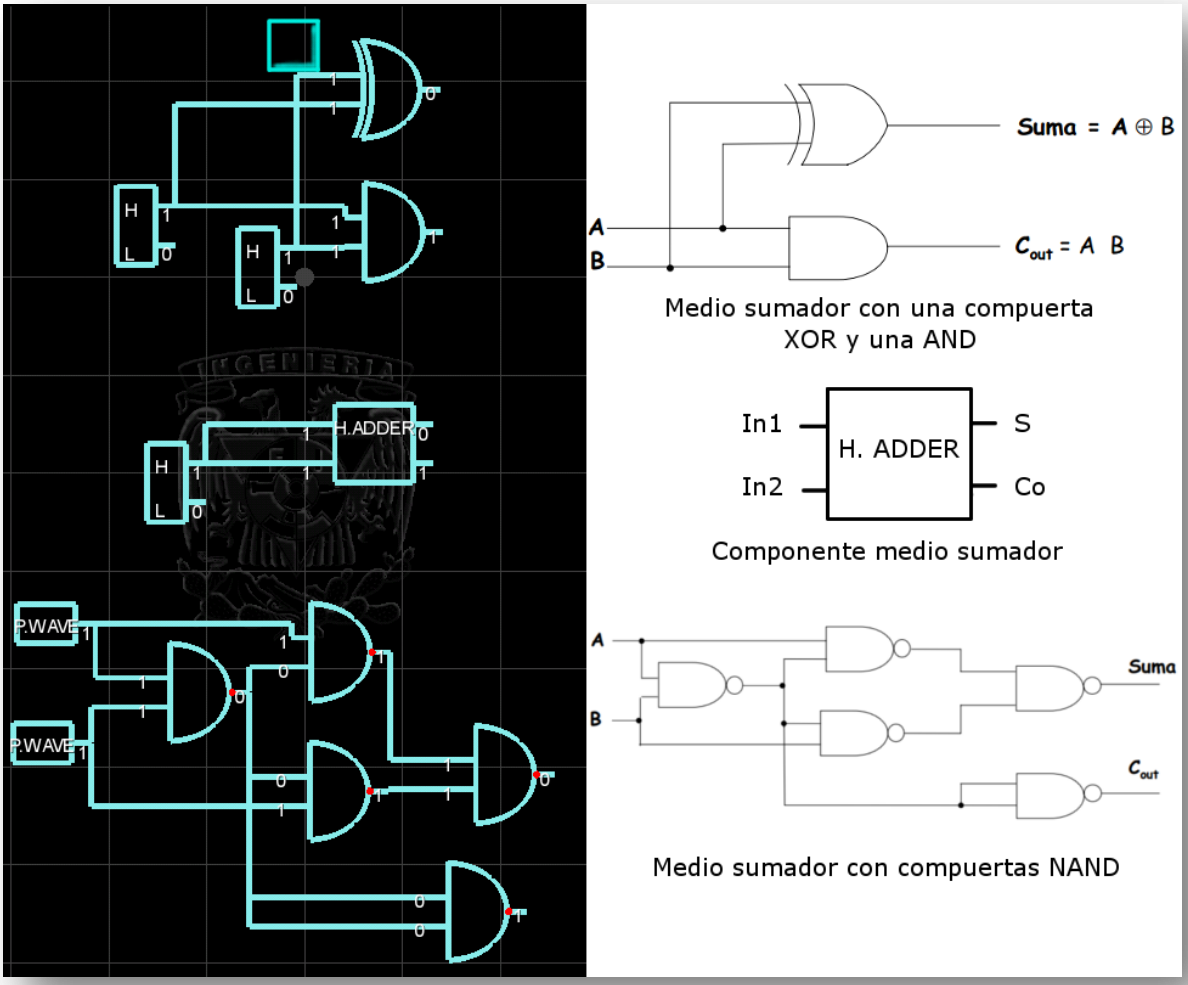


Figura 55: Ejercicio de ejemplo 2

A continuación se presenta el uso del componente comparador de 2 bits. Los pines con la letra A representan los bits del primer número a comparar y aquellos con la letra B los bits del segundo número. Uno de los pines de salida se activará con un nivel alto (1) indicando si el numero A es mayor, igual o menor al número B.

En el ejemplo los pines A tienen un valor 1 representando un número binario tal que $11_2 = 3_{10}$ y los pines de B al no estar conectados se considera un valor 0. Esto significa que A > B por lo que el primero de los pines de salida debe activarse.

Figura 56: Ejercicio de ejemplo 3

El último de los ejemplos muestra un decodificador de BCD - 7 segmentos y un display. El decodificador convierte un número binario en una serie de bits que activan cada uno de los segmentos de un display 7 segmentos mediante los cuales se representa un número decimal.

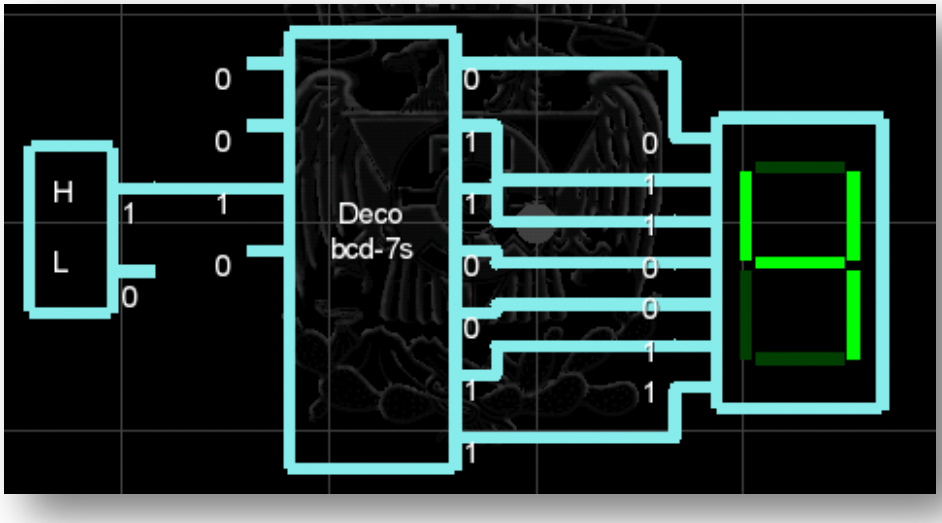


Figura 57: Ejercicio de ejemplo 4

10.1 Versiones de escritorio

En su versión para sistemas operativos de escritorio la aplicación se presenta como un archivo ejecutable de java con extensión .jar que no requiere de instalación. Su ejecución está limitada a que el usuario tenga instalada en su computadora alguna versión de Java.

10.2 Versiones móviles

En su versión móvil la aplicación está disponible como un archivo con extensión .apk el cual es únicamente instalable en un dispositivo con sistema operativo Android. Este archivo permite la ejecución de la aplicación tras su instalación en el dispositivo.

GLOSARIO

Assets: Recursos de diferente índole para el desarrollo de un proyecto.

Código nativo: Estructura de código utilizada para programar en un sistema operativo en específico y que no es compatible con otros sistemas operativos.

Ejecución en tiempo real: Ejecución de un programa en donde los cálculos que se utilizan para presentar la información al usuario se realizan mientras el programa se está ejecutando.

Google Developer ID: Nombre de usuario que se le asigna a una persona una vez que se registra como desarrollador de algún producto de Google.

Marketplace: Tienda virtual de aplicaciones mediante las cuales se puede descargar software o aplicaciones para un sistema operativo ya sea de forma gratuita o por pago.

Smartphone: Dispositivo electrónico móvil con funcionalidad diversa, incluida la de teléfono, que funciona bajo un sistema operativo móvil como Android, iOS, Windows 8, etc.

Software/Programa multi plataforma: Todo programa computacional móvil o no que se encuentra disponible en versiones ejecutadas en distintos sistemas operativos.

Splash screen: Pantalla representativa de un programa que se muestr al iniciar la ejecución de éste.

Versión de escritorio: Versión de un programa computacional que está limitado a poder ser ejecutado en sistemas operativos para computadoras que no tengan cualidades móviles, es decir, de escritorio.

Web app: Aplicación que se ejecuta en un navegador web.

BIBLIOGRAFÍA

“Fundamentos de sistemas digitales”, Thomas L. Floyd
Ed. Prentice Hall, 2006
Pags. 1024

“Taller de Cómputo Descubre Construyendo”, Jara Castro, Sandra - Pérez Mora, óscar
Ed. Umbral, 2005
Pags. 223.

“Sistemas de información gerenciales”, Amaya Amaya, Jairo
ECOE Ediciones, 2010
Pags 228.

“Java 2 – Curso de programación”, Ceballos, Fco. Javier
Coedición: Ra-Ma, Alfaomega, 2006
Pags 880.

MESOGRAFÍA

“Etimología de COMPUTACIÓN” Origen de las Palabras, 2004 <http://etimologias.dechile.net/?computacio.n> (21 Jun. 2014)

Wikipedia, “Generaciones de computadoras” Junio 2014 <http://es.wikipedia.org/wiki/Generaciones_de_computadoras> (21 Jun 2014)

Computación Aplicada al Desarrollo SA de CV, “Generaciones de las Computadoras” <http://www.cad.com.mx/generaciones_de_las_computadoras.htm> (21 Jun 2014)

Labcenter Electronics “The VSM Advantage” <http://www.labcenter.com/products/vsm/vsm_overview.cfm> (21 Jun 2014)

Wikipedia, “Proteus (electrónica)” Junio 2014 <http://es.wikipedia.org/wiki/Proteus_(electrónica)> (21 Jun 2014)

Wikipedia, “Xilinx” Junio 2014 <http://es.wikipedia.org/wiki/Xilinx> (21 Jun 2014)

Creative Commons <http://creativecommons.org/> (21 Jun 2014)

UX matters <http://www.uxmatters.com/index.php> (21 Jun 2014)

LibGDX <http://libgdx.badlogicgames.com/> (21 Jun 2014)

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

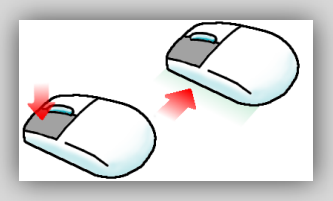
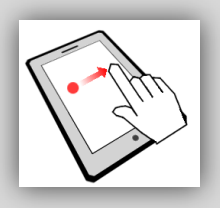
ANEXO 1:
Manual de uso de la aplicación

1. Conociendo la interfaz

Circuit 10 cuenta con una interfaz simplificada que consta de un lienzo de edición y de 3 botones principales: Componente, Play y Eliminar.

Área de edición

El área de edición es el espacio en el que se pueden agregar, suprimir y editar componentes para el armado de un circuito virtual. Abarca la totalidad de la pantalla y se caracteriza por ser un área de color negro con una cuadrícula gris.
En esta área se pueden realizar traslaciones, acercamientos y alejamientos de cámara para visualizar a los elementos que se encuentren distribuidos en ella. Dichas acciones se realizan mediante la interacción con la aplicación de diferente manera según del tipo de dispositivo desde donde se esté ejecutando.

Acción	Equipos de escritorio (mouse)	Dispositivos móviles (pantalla táctil)
Traslación	Click & drag* 	Touch & drag* 
Acercar/Alejar	Scroll de mouse 	Pinch  (pellizco con dos dedos)

*El click/touch para traslación se debe realizar en un punto donde no haya traslape con algún componente.

Botón Componente. 

El botón componente despliega un el *Menú Familias.* Seleccione esta opción cada que desee agregar una nueva pieza al área de edición para revisar las diferentes familias de componentes disponibles.

Botón Play/Stop. 

El botón Play inicia la simulación en tiempo real del circuito o circuitos que se encuentren en el área de edición. El botón Play cambiará de ícono al de Botón Stop para detener la simulación

Botón Eliminar. 

El botón eliminar permite suprimir componentes y conexiones en el área de edición

Botón Rotar. 

Rota la pieza seleccionada 90 grados en sentido anti horario.

Botón Configurar. 

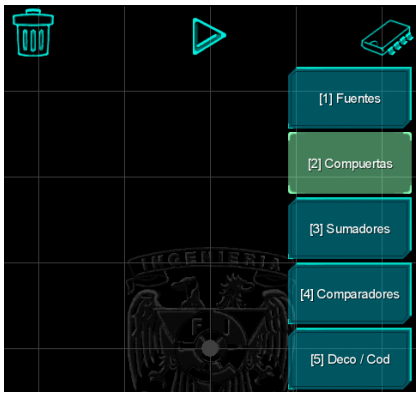
Abre la pantalla de configuración de componente para editar los campos configurables de la pieza seleccionada.

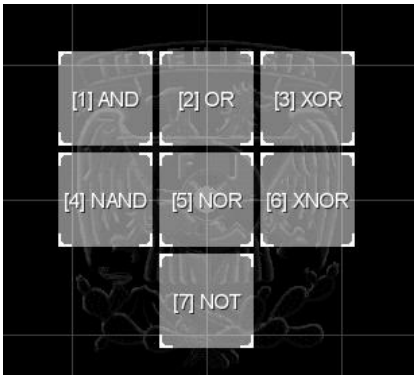
2. Agregando elementos

Para agregar un nuevo componente al área de edición presione el *Botón Componente*, seleccione una de las opciones del *Menú Familias* y luego el componente deseado del *Sub Menú Componentes.*

Menú Familias.

Presione el *Botón Componente* para abrir el *Menú Familias*. Éste menú contiene un listado seleccionable de las diferentes familias de componentes disponibles en la aplicación. Al seleccionar una familia se abrirá el *Sub Menú Componentes.*





Sub Menú Componentes.

El *Sub Menú Componentes* contiene todas las piezas disponibles dentro de la familia seleccionada en el *Menú Familias.* Al seleccionar una de las opciones disponibles en este menú se agregará un elemento gráfico al área de edición correspondiente a la pieza seleccionada.

3. Realizando conexiones

Nueva conexión

Puede crear conexiones entre componentes seleccionando los pines de los mismos. Para crear una nueva conexión de click o toque un pin de origen de la conexión y posteriormente un segundo pin de destino. El pin de origen se marcará con un círculo amarillo para indicar que fue correctamente seleccionado mientras que el segundo pin, si fue correctamente seleccionado, se unirá al primero mediante una línea.

NOTA: Las conexiones solo pueden ser realizadas entre pines de entrada y salida, de lo contrario el programa lanzará un mensaje de advertencia.

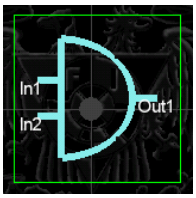
Selección de conexión

Se pueden seleccionar líneas de conexiones haciendo click/touch sobre ellas. Una conexión seleccionada cambiará su color a amarillo. Esto también permite diferenciar conexiones cuando se encuentran en un área muy saturada.

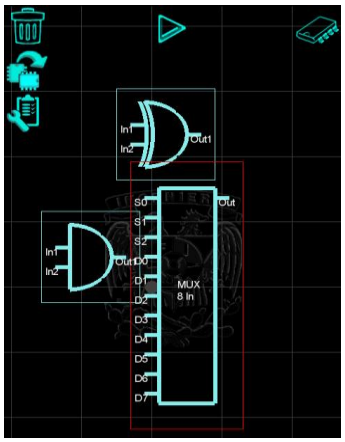
4. Modificando elementos

Seleccionando componentes

Para seleccionar un componente de un click/touch sobre la pieza que quiera seleccionar del área de edición. La pieza seleccionada se rodeará por un marco de color verde que indica que esa es la pieza que se tiene seleccionada actualmente.
La selección de una pieza activa los botones de *Rotar* y *Configurar*.



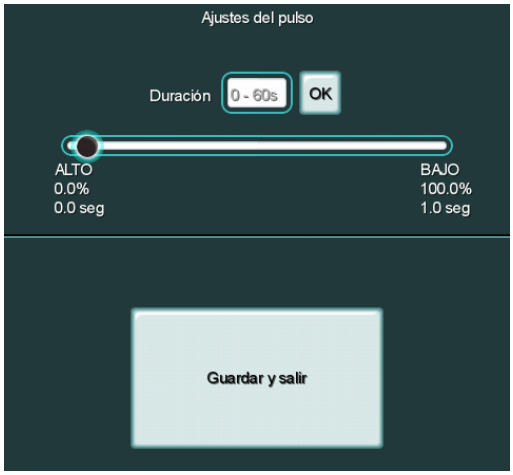
Reubicando componentes

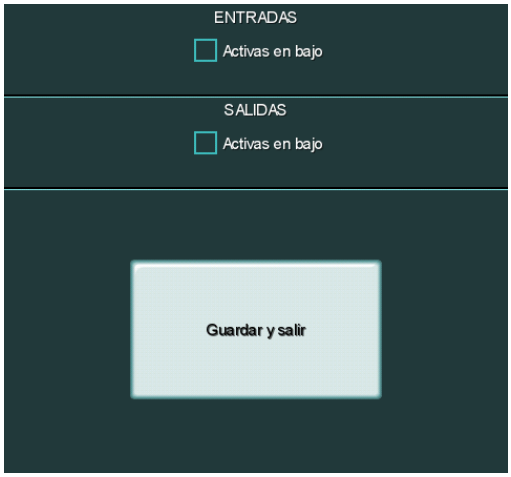


Puede reubicar un componente dentro del área de edición haciendo click/touch sobre él y arrastrando el click/touch sin soltarlo hasta a la ubicación deseada. El componente que se esté reubicando se rodeará con un marco verde indicando que la nueva posición es válida. El marco cambiará a color ROJO cuando haya traslape entre el componente a reubicar y las piezas existentes en el área de edición. Si la posición final no es válida el componente regresará a su última posición válida.
En el caso de la rotación la pieza no podrá girar si no hay suficiente espacio.

Configurando un componente

Seleccione el *Botón Configurar* para modificar las características del componente seleccionado. Al presionar la opción configurar se cambiará a la *pantalla de configuración*, la cual muestra una lista con todas las opciones configurables disponibles para el componente en cuestión. Para regresar a la pantalla del área de edición seleccione en el botón *Guardar y salir* ubicado al final del listado de opciones de configuración.





5. Eliminando elementos

Se pueden eliminar componentes y conexiones del área de edición.

Supresión de componentes

Para suprimir un componente existente en el área de edición se debe primero seleccionar dicho componente y posteriormente presionar el *Botón Eliminar.* Si el componente tiene conexiones conectadas éstas también serán eliminadas.

Supresión de conexiones

Para suprimir una conexión existente en el área de edición se debe primero seleccionar la conexión deseada y posteriormente presionar el *Botón Eliminar.*

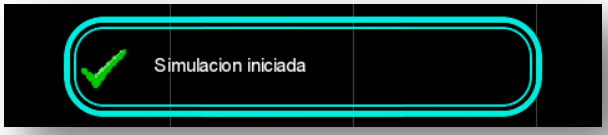
6. Simulación

La simulación es el despliegue en tiempo real de los valores presentes en los pines de entrada y salida de los componentes del área de edición.

Iniciando

Para iniciar la simulación de el o los circuitos presentes en el área de edición basta con presionar el *Botón Play*.

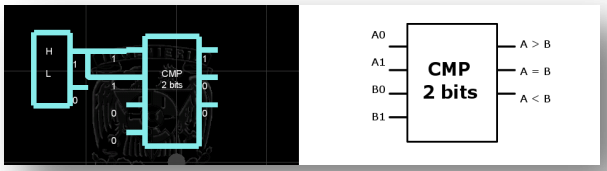
Al entrar en modo de simulación se mandará un aviso en pantalla con la leyenda "Simulación iniciada", el botón Play desaparecerá y tomará se mostrar el *Botón Stop.*



Durante la simulación.

Durante la simulación los nombres de los pines de todos los componentes desaparecerán y en su lugar se mostrarán 1s y 0s según el valor que éstos estén arrojando. Los botones Componente y Eliminar desaparecerán por lo que no se podrán suprimir componentes o conexiones existentes ni agregar nuevos.

El control de la cámara seguirá habilitado para que el usuario pueda recorrer el área de edición y visualizar diferentes áreas del circuito al momento de la simulación.



Deteniendo

Para detener la simulación basta con presionar el *Botón Stop* durante la simulación. Los pines de las piezas dejarán de mostrar su valor binario y recuperarán su nombre correspondiente.

7. Componentes disponibles en esta versión

Los componentes disponibles en la aplicación están separadas por familias de funcionalidad común.

Fuentes

- Clasica (valor constante)
- Tren de pulsos

Compuertas (2 bits)

- AND
- OR
- XOR
- NAND
- NOR
- XNOR
- NOT

Sumadores (2bits)

- Sumador completo
- Medio sumador

Comparadores

- 2 bit
- 4 bit

Decodificadores/Codificadores

- Deco 2 a 4
- Deco bcd a dec
- Deco bcd a 7 segmentos

- Cod 4 a 2
- Cod dec a bcd

Multiplexores/Demultiplexores

- Mux 1 de 4
- Mux 1 de 8
- DMux 1 de 4
- DMux 1 de 8

Latch

- Tipo SR
- Tipo SR (habilitador)
- Tipo D

Flip flop

- Tipo SR
- Tipo D
- Tipo JK

Display/LED

- Display 7 segmentos
- LED