UNIVERSIDADE DE ÉVORA

Escola de Ciências e Tecnologia

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

"The X in balance", a serious game to learn how to solve mathematical equations

Yusra Tehreem

Orientação: Francisco Manuel Gonçalves Coelho

Mestrado em Engenharia Informática

Dissertação

Évora, Julho 2016

UNIVERSIDADE DE ÉVORA

Escola de Ciências e Tecnologia

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

"The X in balance", a serious game to learn how to solve mathematical equations

Yusra Tehreem

Orientação: Francisco Manuel Gonçalves Coelho

Mestrado em Engenharia Informática

Dissertação

Évora, Julho 2016

This work is dedicated to my father Zafar Ali, who taught me mathematics when there were no such games in my childhood that can act as a teaching support tool. My father made me to love maths, now I am trying to made all students love maths.

Preface

This dissertation is submitted for the degree of Masters (Engenharia Informática) at University of Évora. Under the supervision of Professor Francisco Manuel Gonçalves Coelho, i have selected to work on game design. With the specific period of time and resources, an attempt has been made to make a serious educational game. While writing this thesis, the objective was to describe a math game for solving mathematical equations. Injecting learning factor in a game, is a main concern of this project. The document is about the description of 'X in Balance' game. This game provides a platform for school aged students to solve the equations by playing game. It also gives a unique dimension of putting fun and math in a same platform.

The document describes full detail on the project. The first chapter gives an introduction about the problem faced by students in doing maths and the learning behavior of a game. It also points out the opportunities that this game might brings and the motivation behind doing this work. It describes the game concept and its genre too. Besides, the second chapter tells state of an art of serous educational game. It defines the concept of serious game and its types. Furthermore, it justifies the flexibility of serious games to adapt all learning styles. The impact of serious games on learning is also mentioned. It also includes the related work of other researchers.

In the thesis, third chapter describes objective and features of a running game. This section gives a detailed description of modular structure, made for developing a game. It allows full technical and formal description of game with describing testing details. The last Chapter concludes the whole project. It also gives future dimensions of the developed game. The main chunks of code behind the running game are also mentioned. Besides, the demonstration of game play is described in the end of this document.

Acknowledgment

Above all, I am thankful to Allah Almighty (The God) from selection of this particular project to the successful efforts made for developing an educational game.

I would give special thanks of gratitude to my supportive supervisor Francisco Manuel Gonçalves Coelho, who gave me an opportunity to do this interesting project on game design. It helped me a lot in exploring new dimensions specially designing game as a learning tool. I learned so many new things from programming enhancements to complex game design and i am really thankful to him.

I would also like to thanks my family and friends for their prayers and cooperation with me during the period of development. Besides, i would acknowledged my university professors who taught me different courses. Because of them, my concepts and skills were groomed and i was able to do this work.

Contents

List of Figures

List of Tables

Acronyms

URL *Uniform Resource Locator*

IIFA Instituto de Investigação e Formação Avançada

ECT Escola de Ciências e Tecnologia

UE Universidade de Évora

SEG Serious Educational Game

PNUM Positive Number

NNUM Negative Number

PVAR Positive Variable

NVAR Negative Variable

PVARC Positive Variable with Coefficient

NVARC Negative Variable with Coefficient

ITM Item

OPR Arithmetics Operators

SUM Addition

SUB Subtract

MUL Multiply

DIV Divide

Abstract

Integration of learning with digital technologies could make a powerful education system. Nowadays, students become easily distracted with regular teaching methods. It does not mean that its a problem of these methods, its actually due to the attraction, that a technology brought in the form of digital applications and games. Thus, these books, black boards and lectures could not get the attention of students in front of fascinating ubiquitous devices. With the problem of distraction, learning is also becomes hard. Education always had this burden, augmented with various economical, social and cultural restrictions. Students and teachers face increased difficulties from the conjunction of regular teaching methods with the complexity and novelty of course materials. A poor learning environment not only makes an hard task harder but also lowers the motivation which eventually leads to the abandonment and rejection of learning.

Mathematics is a clear subject of this situation: abstract concepts are hardly understood by students. The result is that many have long standing difficulties with the conceptual tools used in solving mathematical problems. This quickly evolves into stress and later into poor grades, illiteracy and negative reactions to mathematical and scientific content.

Considering above situation, the project is to develop a serious game. Its purpose is to teach a specific mathematical method: Solving equations in a form of fun. The game is based on solving mathematical equations by balancing the variable 'x' (in equation). It is basically, finding a value of x to make equation balanced.The resolution of an equation can be conveyed as keeping a balance balanced. The player is presented with an equation and a set of operations. It must choose one operation, that will be applied to both sides of the equation, producing a new (equivalent) equation and new operations. This process is repeated until the equation has a form where the unknown variable "x" is isolated in one side and the other side evaluates to a number.

In this game, a modular structure is presented that contains seven modules. Starting from level selection, in which the type of equation will be selected for solving. Second one is representing selected equation in mathematical form so that a player clearly sees it, like in a paper. The third module generate operations/options to apply on equation while the fourth module apply the selected option. The core of a game lies in module called simplification. It includes an approach of simplifying equation by checking and apply several sets of basic math rules. Besides, history management is also taken care of doing undo and redo on the equation states. The last module have taken care of score management from selection of right and wrong options. These all modules combine to produce a playable math game.

The project has done the implementation of a single-player playable prototype. It is capable of solving

single variable equation of degree one. The player competes to get higher score percentage and is able to see the mistakes made during resolution of an equation in a game. This work settled the design guidelines and the structure of a framework to enable further developments such as: multi-player modes (cooperative or competitive), larger coverage of mathematical contents (trigonometry, second-order equations, linear systems, etc) and many victory conditions (faster or shorter resolution).

Keywords: Mathematics, Practice Tool, Serious Game, Online Teaching, Competitive Learning, Educational Game

Sumário

O X na Balança

Um jogo sério para aprender a resolver equações matemáticas

A integração das tecnologias digitais na aprendizagem permite sistemas de ensino poderosos. Atualmente os alunos distraem-se facilmente com os métodos de ensino regulares. Isso não significa que haja um problema nestes métodos, na realidade ocorre devido à atração que a tecnologia trouxe sob a forma de aplicações digitais e jogos. Assim, os livros, quadros e lições tradicionais não podem conquistar a atenção dos estudantes frente aos fascinantes dispositivos omnipresentes. Com o problema de distração, a aprendizagem é também se torna difícil. O ensino teve sempre esse fardo, aumentado com as várias restrições económicas, sociais e culturais. Alunos e professores enfrentam dificuldades acrescidas na conjunção dos métodos de ensino regulares com a complexidade e novidade dos materiais letivos. Um ambiente de aprendizagem pobre não só torna uma tarefa difícil mais difícil, mas também reduz a motivação o que eventualmente leva ao abandono e rejeição de aprendizagem.

A matemática é uma exemplo claro dessa situação: os conceitos abstratos são mal compreendidos pelos alunos. O resultado é que muitos têm dificuldades persistentes com as ferramentas conceituais usadas na resolução de problemas matemáticos. Isto evolui rapidamente para a saturação e mais tarde para más classificações, iliteracia e reações adversas aos conteúdos matemáticos e científicos.

Considerando a situação acima, propomo-nos desenvolver um jogo sério. A sua finalidade é ensinar um método matemático específico: resolver equações de uma forma lúdica. O jogo inspira-se na resolução de equações matemáticas, equilibrando a variável 'x' (na equação). Basicamente, consiste em encontrar um valor de x que equilibre a equação. A resolução de uma equação pode ser descrita como manter uma balança em equilíbrio. Ao jogador é apresentada uma equação e um conjunto de operações. Este deve escolher uma operação, que vai ser aplicada a ambos os lados da equação, produzindo uma nova equação (equivalente) e novas operações. Este processo é repetido até que a equação tenha uma forma em que a incógnita 'x' é isolada de um lado da balança e no outro lado está um número.

Neste jogo, é apresentada uma estrutura modular com sete módulos. Inicialmente há uma seleção de nível, onde é escolhido o tipo de equação a resolver. De seguida é apresentada uma equação selecionada do tipo escolhido, em forma matemática para que o jogador possa lê-la claramente, como numa folha de papel. O terceiro módulo gera operações/opções para aplicar à equação enquanto o quarto módulo aplica a opção

selecionada. O núcleo de um jogo reside no módulo de simplificação. Este módulo inclui um processo de simplificação da equação, verificando e aplicando vários conjuntos de regras básicas de matemática. Além disso, a gestão do histórico também é tratado, permitindo desfazer e refazer passos na resolução da equação. O último módulo trata da gestão pontuação a partir da escolha de opções certas e erradas. Todos estes módulos combinam-se para produzir uma experiência jogável de um jogo de matemática.

O projeto foi realizado através da implementação de um protótipo jogável para um jogador. Este protótipo é possível resolver equações do primeiro grau com uma única variável. O jogador compete para obter a maior pontuação percentual e pode ver os erros cometidos durante a resolução de uma equação. Este trabalho estabelece a estrutura de desenho e base de um quadro de forma a permitir novos desenvolvimentos, tais como: modos multi-jogador (cooperativo ou competitivo), maior cobertura dos conteúdos matemáticos (trigonometria, equações de segunda ordem, sistemas lineares, etc) e várias condições de vitória (por exemplo a resolução mais rápida ou mais curta).

Palavras chave: Matemática, Ferramenta de Prática, Jogo Sério, Ensino Online, Aprendizagem Competitiva, Jogo Educativo

1

Introduction

In the field of education, various challenges are coming. Due to the complex and advance course material, students face difficulties along with their teachers to grasp all the concepts by regular teaching methods. It causes burden and convert it into boredom. Poor learning environment not only makes an hard task harder but also lowers the motivation which eventually leads to the abandonment and rejection of learning. The subject of mathematics clearly explains this situation. The broad and critical concepts could not easily grasp by all the students.Most of the students have long standing difficulties with the conceptual tools used in solving mathematical equations. This problem quickly evolves into boredom and later into poor grades of mathematics. Although, the tools have progressed a lot and finding new ways in every field but it is also needed to change the traditional teaching method by the help of advance technologies in computer sciences.

Most students enjoy playing computer games. The use of games as a teaching support tool is not novel and indeed, it is an active field of research [Mac13] [KC10]. One of the main advantages of well designed serious games is the presentation of the content in a way that stimulates the human natural curiosity and search for fun, instead of "covering the broccoli with chocolate". Games are, perhaps, the most convenient medium to produce an effective hands-on learning experience [TFDW11] [VVCB$^+$06].

Our motivation is to develop a serious game with the purpose of adding the human driven search for fun to learn a specific mathematical method: Solving equations. Keeping a balance balanced is a well-known metaphor to equation solving and this image lends itself to the development of a simple game core (for example, the use of addition and/or subtraction of terms in a simple linear equation or the application of the distributive law in a subexpression) while opening the possibility of many game experiences.

With that motivation in mind our goal with this work is the implementation of a single-player playable prototype, eventually polished by the outcomes of play-testing. Furthermore, this work also aims to settle design guidelines and the structure of a framework to enable further developments such as: multi-player modes (cooperative or competitive), larger coverage of mathematical contents (trigonometry, second-order equations, linear systems, etc) or many victory conditions (faster or shorter resolution).

1.1 Motivation

Now a days, students are growing up in a digital World that is completely different and advance as that of their classrooms. There are traditional teaching methods like text books, black boards, lecturing to the students. As the students are now always in front of computers, mobiles and televisions. So these traditional teachings could not catch the attention of the students [HCSB11]. Thus the traditional classroom is loosing the battle for the attention of students. Books, blackboards, lectures and exercise resolution must compete with loud and colorful television shows, adrenaline boosting computer games and immediate demands from "social networks" in ubiquitous mobile devices. Besides, we could not say that a mathematical game can replace a teacher but it can act as an assistance system to increase student skills [TBF15] [Yan12]. A fun (as understood by players) mathematical game could set a baseline for the development of more mathematical games. One solution to cope with the abstraction problem in mathematics is the acquisition of familiarity through the resolution of many exercises. This apparent difficulty turns out to be a common feature in many addictive games. For example flappy bird and 2048 game, where a small set of mechanics is used to overcome increasingly difficult scenarios. Likewise, solving equations (and other mathematical methods) rests in a small set of rules that can be applied in increasingly difficult problems.

Besides, mathematics do not only need understanding but also practicing. Some students found difficulty in understanding the concept while others are too lazy to practice the problems on paper. Thats why many students hate mathematics. But the good thing is student loves game, so to put learning inside the game could be very productive. Thus another motivation is to inject fun to learn and practice mathematics in the form of game.

1.1.1 Opportunities

The development of fun mathematical game is a small but an effective step towards a resolution of some challenges that learning and teaching face today. Below are the main opportunities that such a game might bring.

- **Introduction of digital media in a classroom:** This "novelty" in traditional teaching methods places familiar objects, often associated with fun, in the center of the learning process.
- **Social motivation and Practicing Medium:** A new method of teaching is formed that deals with the psychology of students and allow them to solve mathematical equations by their own interest. The motivation for wining the game, makes students to solve as much equations as they can. In this way, the game will provide an effective medium for practicing mathematics.

- **Extensibility:** A good set of guidelines and a general framework facilitates further development of new modes of play (for example, multi-player, against the clock, etc) or new syllabus content (for example, trigonometry or second order equations, linear systems, etc).

1.2 Game Concept of 'X in Balance'

Game is not used for fun only. Through the medium of fun, one can achieve better learning effects [PMP10]. By developing the mathematical game , the goal of teaching mathematics can be achieved. The game is based on solving mathematical equations by balancing the variable 'x' (in equation). It is basically, finding a value of x to make equation balanced. The algorithmic resolution of an equation rests in a process of simplification that iteratively transforms the equation (using well defined, equality preserving rules) until the unknown is isolated in one side and the other side evaluates to a number. The equality preserving rules can be intuitively conveyed using the balance metaphor: whatever is done to one side of the balance/equation must also be done to the other side so that, if the balance/equation was balanced/true before the operation then after the operation remains balanced/true. Thus, solving an equation becomes a problem of selecting the "right" operations that will be applied to both sides. The 'X in Balance' game concept focuses in this selection. The player is presented with an equation and a set of operations. It must choose one operation, that will be applied to both sides of the equation, producing a new (equivalent) equation and new operations. This process is repeated until the equation has a form where the unknown is isolated in one side and the other side evaluates to a number.

This simple concept can be instantiated in many game experiences. For example, a good choice of the operation leads to a simpler equation but a bad choice makes the equation more complex. Thus, the presentation of operations (options, in game design language) can be used to drive the game experience and with it, the learning process. Also, a particular resolution can be scored by time (the fastest the better), by number of operations (the shortest the better) or by other criteria. Again, (automatically) scoring resolutions can be used to reach learning goals through game play.

Figure 1.1: 'X in Balance' - Getting Value of 'X' While Keep the Equation Balanced. The figure shows an image behind concept

1.2.1 Classification

The given game is classified into Serious game category. Serious games design for specific purpose other than pure entertainment. Usually, a game is classified by its perspective, mechanics, dynamics or play pattern like traditional games do i-e "First Person Shooter" [BS09]. But serious game is classified by purpose or specific goal. So the primary purpose of current serious game is to learn concepts of mathematics and then to entertain the students. Thus the given game 'X in Balance' is a Serious Educational Game (SEG). It can also be called Edutainment game (Educational entertainment) that is entertaining but its primary purpose is to educate. The figure 1.2 shows this classification clearly.

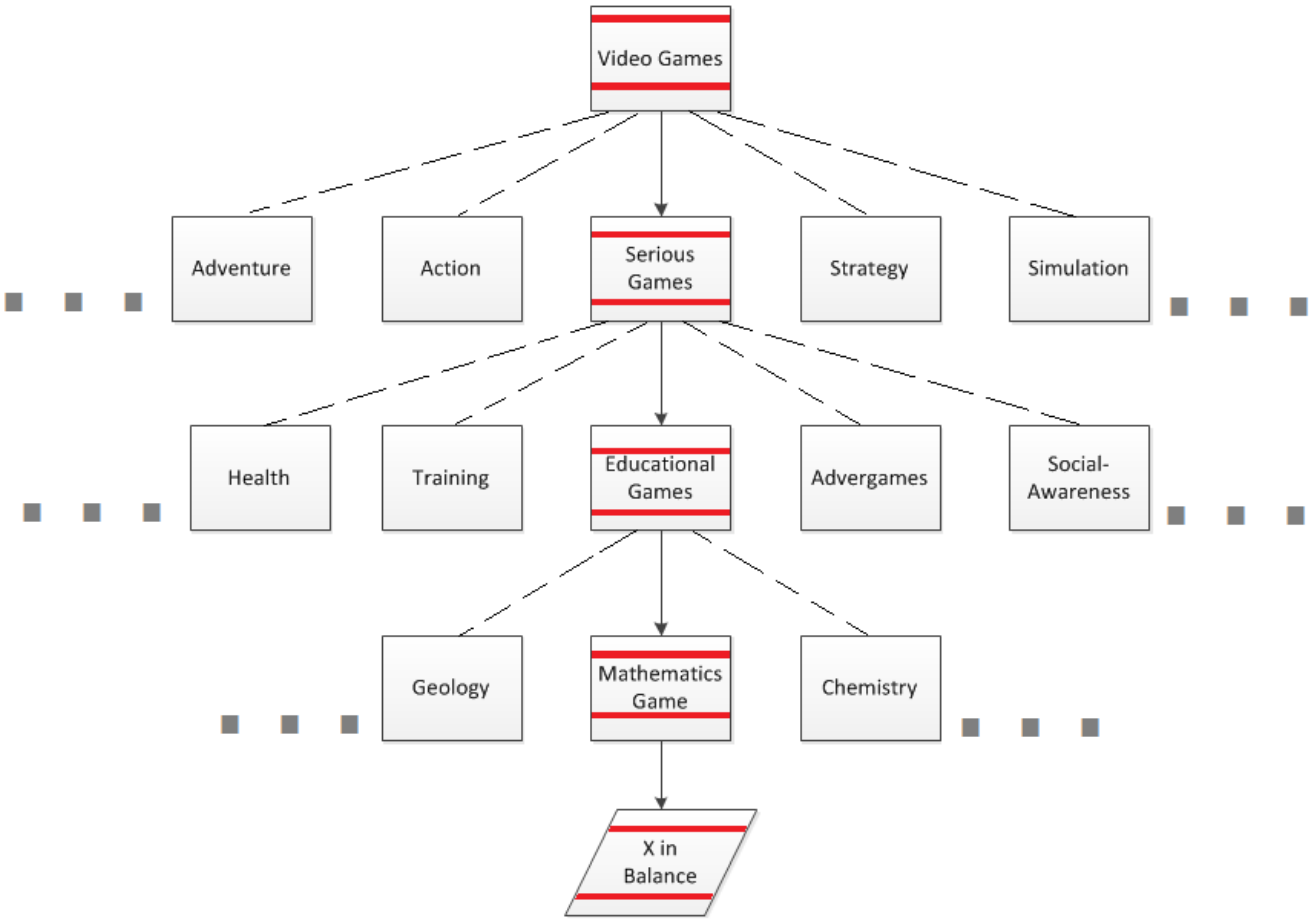

Figure 1.2: Classification of a Developed Game in Family Tree of Video Games

2

State of Art

There are various concepts in making computer games but the main goal is to entertain, after all it is a game. But when we look at serious game, it is different and purposeful. Serious game is more than just pure entertainment. For example a book not only gives information but persuades, educate, and give a call to act upon it. Similarly a game that goes beyond fun and entertainment is called 'serious game' [BS09] [BAAH13]. Educational game is a sub category of serious game with purpose of learning. Thus we can say that serious game has some purpose and if the purpose is learning then it is called Serious Educational game. The developed game 'X in Balance' lies in this category. Thus it is necessary to first acquire knowledge about serious and educational game. There are few notable definitions of serious games by several researchers. Zyda [Zyd05] states:

"Serious game: a mental contest, played with a computer in accordance with specific rules, that uses entertainment to further government or corporate training, education, health, public policy, and strategic communication objectives."

Although it covers almost all aspects but its entertainment factor sometimes clashes with the core of serious game as defined above. Considering this point, Backlund et al [BH13] defines serious games as:

"Games that engage the user, and contribute to the achievement of a defined purpose other than pure entertainment (irrespective of whether the user is consciously aware of this). A game's purpose may be formulated by the user her/himself or by the game's designer, which means that also a commercial off-the-shelf (COTS) game, used for non-entertainment purposes, may be considered a serious game."

Thus a game beyond entertainment can be included in it.

2.1 Forms of Serious Games

Some researchers have tried to obtained a generalized forms/labels instead of fixed categories as explained by Breuer et al [BB10]. Thus we could not say that there are fixed types of serious game. There are several forms/types of serious games according to Brathwaite and Schreiber [BS09], listed below. It is not hard and fast listing. Rather, these are some possible ideas that can make a serious game.

- **Training Games:** It is designed to put a player in a specific situation and let him/her to achieve mastery and comfort over it before doing it in a real world. Cantoni et al [CK10] designed a hospitality training game called 'Waiter Game' to support the young students in school of professional hospitality.
- **Health Games:** It is designed for two purposes. One is to train people to make health related activities in several situations [GSB14] and the second one is to make player healthier. The best example is 'Re-Mission 2' [rem]. It is for kids and young adults having cancer. It puts a player into a human body and allow a player to fight with cancer with some weapons like chemotherapy, natural defenses and antibiotics. The game will be played parallel to real world activities for destroying cancer and win.
- **Social-Commentary Games**: It is designed for spreading a point across the World through internet. One of interesting example is 'Airport Security' [air] in which the game allows a passenger to explore restrictions in American airports. Similarly political games also lie in this category. In election time, a party releases a game to show its policies and also mistakes of their opponents.
- **Advergames and Anti-Adver Games:** The games that are designed for advertising a brand [APD14] or actively damage a reputation of some corporation. 'Big Bumpin' is an advergame for Burger King [Wik]. McDonald's Game by Molleindustria [mcv] is a kind of anti-adver game. It allows player to explore dirty secrets that make McDonalds a biggest company. It has no connection with McDonalds itself.
- **Educational Games:** Similar to training games, but educational games focus on school age players. These games have ability to teach. The content of subject is presented in such a way that encourages active learning. For example the current developed game 'X in Balance'.
- **Social-Awareness Games:**It is designed to put a player in an uncomfortable and annoying situation in order to aware the player about real life happenings. 'Dying for Darfur' [dar] is an award winning game in this aspect. The primary goal of a game is horrible i-e run to escape from getting hurt, theft, raped or worse. Another game EnerCities is also an example of this form i-e environmental awareness game [SMP15].

2.2 Flexibility of Serious Educational Games and Learning Styles

There are different types of students in the class. Some are active and some behave passively. The active students like to answer questions and perform a lead role in group study. Usually they participate in

the classroom. While passive students only listen the instructor and usually shy to do anything in the classroom. Thus it is very important to present teaching material in a stimulating and interactive way. Different students have different learning styles in which they learn effectively. There are three commonly known styles of learner listed below and the methods used in these styles [Leo12] are presented in table 2.1.

- **Visual Style Learner:** Such type of learners learn effectively when they see anything like body language or facial expression of teacher.
- **Auditory Style Learner:** Such learners found easiness by aural means like discussion, verbal lectures and other talking things.
- **Kinesthetic Style Learner:** These type of learners understand well through some physical activities, moving and touching involvements.

Table 2.1: Materials and Activities Used in Learning Styles

Visual	Auditory	Kinesthetic
Posters	Debate	Role-Plays
Pictures	Dictation	Drama
Drawings	Story Telling	Movement
Table	Reading	Handling props
Graph	Lectures	Races/other competitions

The good fact is educational game has the ability to allow these all three styles as shown in figure 2.1. The combination of images, text and animations make a game suitable for visual learner. Besides, adding sound to describe images or other content make also a game appropriate for auditory learners. Above all, by using mouse, keyboard or touch system, can make kinesthetic learners to learn at their best.

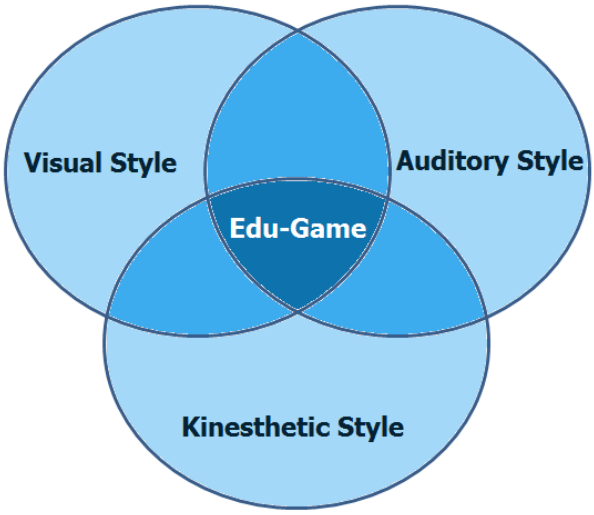

Figure 2.1: Flexibility of Educational Game. The figure shows the ability of edu-game to include all three styles of learning in terms of adding images, sound and input interaction

2.3 Impact of Serious Games

Educational game is a popular subject over a last decade. Many researchers have done their work in this field. Backlund et al [BH13] did a research on efficiency of serious games. They have done meta-analysis on the impact of serious game in last decade and focused on empirical evidence. The study focus on game-based learning and its usage in various school types i-e higher education, high school, secondary school, elementary school and pre-school. They acknowledged and selected 40 main papers from several databases. The research showed that the serious games made a positive influence on learning. In the figure 2.2, 29 out of 40 studies show positive effect of serious game. Besides, seven studies are neutral and two shows negative impact on learning. There are two studies that are unclear about the impact. Thus overall result concluded that a serious game is an effective learning material.

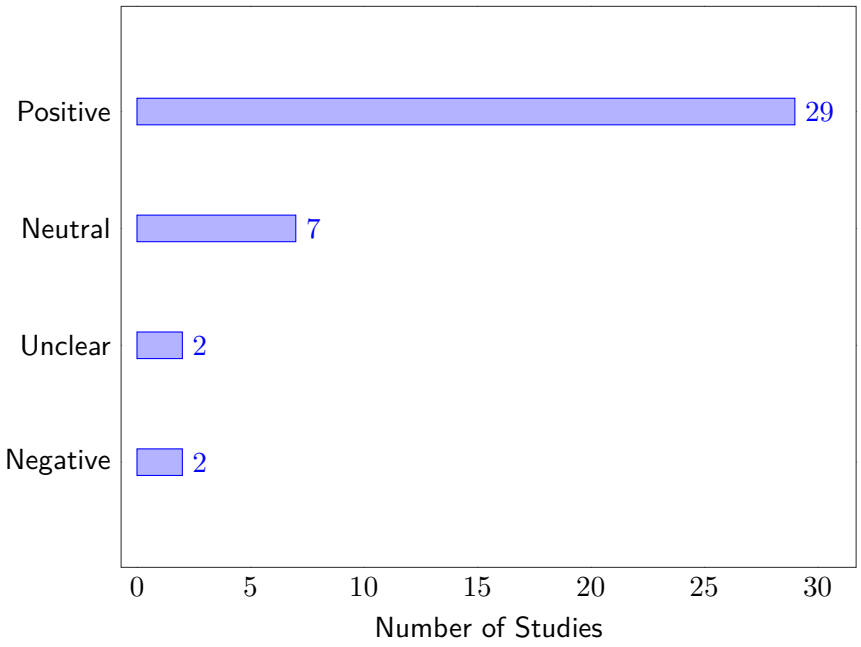

Figure 2.2: Effects of Serious Games on Learning. The graph shows the number of studies that respond positively, neutral, unclear and negatively

2.4 Related Work

There are many researches and applications that are made in the field of serious game and more specifically in educational game. The purpose of educating player leads to many useful applications and approaches.

2.4.1 Math Game Based on Story Telling Approach

Many researchers used storytelling approach in educational game. Giannakos et al [GCJ12] proposed a storytelling method in a video game to learn mathematics. They used Scratch for developing the game. The game is named Gem-Game which is also available by Scratch community [scr] for further improvements. The main purpose of a game is to develop mathematical skills. It has a plot which starts from a small story with a mission. This combination of story and mission stimulates the player's interest and keeps the motivation high. They also made the plots and dialogues funny to put interest. The proposed approach

resulted beneficial for students that had poor grades in mathematics.

2.4.2 Assisting Project Manager in a Form of Game

Lino et al [LPB+15] presented the Project Management Game. It was a SEG to assist the software project managers that are inexperienced. The game was designed to train the managers while considering time, cost, human resources and risk. It gave a real environment in which a manager has to pass through all phases of software development life cycle in a form of game.

2.4.3 Math Game Designed for Visually Impaired Students

Another researcher Ferreira [FC14] proposed a game environment that can be used in home and classroom by the students who are visually impaired. A game-based learning approach was adopted for middle school students who study mathematics. The goal is to teach mathematics to blind students and make them to like this course. The graphics of game was intended for students that have low vision while for blind students; it can be playable without graphics. For this, the game was complemented with spatialized 2D audio. The combination of both graphics and audio features makes an integration of low vision students with their blind colleagues.

2.4.4 GPS Based Mathematics Mobile Game

Wijers et al [Zyd05] designed a game called MobileMath. It is a mobile game that supports GPS. The purpose behind building this game was to investigate the engagement of students in learning mathematics through social and advance type of game.

In a game, teams compete to get points by covering as much space as possible on a playing field. It is done by making circles, parallelograms or squares by walking physically and clicking on each point (vertex) with the help of GPS. Management of a space with time, hinder the other's space or deconstruct other's shape will lead to gain points in a game. The locations and quadrilaterals of all the teams are visible on every player's phone. The data is also stored on line to discuss or view later. Results showed that the students learned to read a map, use GPS and how to make different quadrilaterals.

2.4.5 Educational Game to Learn Computer Programming

One more interesting work related to SEG has done. Muratet et al [MTJV09] proposed a serious game that is devoted to strengthen the skills of computer programming. They chose Real Time Strategy genre to build such type of serious game. The game is implemented by modifying the existing RTS engine to allow a collaborative and safe programming activity over an API. The game allows students to command entities of game with their AIs and can have contest with their colleagues through multi player mode. The game can be flexible to specific programming languages.

3

Resolution

The game is about practicing mathematical equations to school-age students in the form of fun and entertainment. The theme of game is learning and practicing topics of mathematics in an efficient way. There are many topics that can be included. For example Algebraic equations, trigonometry, derivation, integration etc. As the main side of this game is to provide base design to add further advance topics in future. Thus going towards initial step, the base of game is designed that is able to solve linear one-variable equations.

The basic platform with design guidelines is developed. It allows a student to play and practice maths. Initially, it is a single player game. Students that learn in the class will have access on game to practice their skills and evaluate their performances.

The game will show a one-variable equation that need to be solved by student. According to provided equation, several options will be given to solve it. These options include multiple correct options and also some wrong options in order to make it a challenging game. The player needs to choose right and efficient option for each step to solve with good score. The selection of wrong option will lead to more lengthy equation and this will increase time and complexity. For choosing right option, player will get positive score while for wrong option, player will get negative marking. So right option at right time will makes a player to excel the game.

3.1 The Statement of Game

3.1.1 Objective

The main objective is to make a student to practice and learn mathematics in an entertaining way because to practice maths on paper always be boring for students. Thus its goal is to increase interest and practice maths in an efficient way. Besides, the game is also designed to put a basic design guidelines in order to develop more advance-topic games in the field of mathematics gaming.

3.1.2 Required Components

For Developer

- Desktop PC/Laptop.
- Course contents of Mathematics.
- Math rendering libraries.
- Internet Access.
- Smart phone/Tablet for testing.

For User/Player

- Desktop PC/Laptop/Tablet/Smart Phone
- Guidelines/Tutorial about basic knowledge of equations.
- Internet Access.

3.1.3 Restrictions

- Not too overloaded with teaching instead of fun.
- Given equation should be univariate. It includes only one type of variable with degree one.

3.1.4 Genre

The game belongs to educational game from types of serious game. The core of game is 'Collection' where the player have to give and match correct answers to pass several levels for high scores.

3.1.5 Target Audience

The target audience of the game includes school age students that are studying maths and wants to be hold on the concepts of mathematics.

3.1.6 Intended Platform

The game refers to web/mobile application game. So it is developed in HTML5 with JavaScript as a programming language. For interfacing and style, CSS and Jquery is also used.

3.1.7 Material to play

- Electronic device (Computer/Tablet)
- Course book (for help)

3.2 Features

The following features would be the essence of the game:

Represent Original Math Problems and their Solution The game contains real information about every topic present in the levels. Their methods of solutions and answers would be according to original school books of mathematics.

Multiple Methods of Solving As describe above, the game will provide options for every problem. the option will contain more than one correct options. So it will enhance the skills of player to solve any problem in multiple correct ways.

Single Playing Option The game provides single player option. The player will play a game in order to get high score i-e high percentage just like in school gradings.

Different Types of Levels The levels are the types of equations from basic to complex. It needs to be solved by the students step by step to learn systematically. Basic level gives learning skill, standard level gives practicing skill and complex level gives excelling skill.

Simplify by Rule-Method The game provides a rule-method for simplification. It contains set of rules to solve the expressions in an equation. This rule-method act as a base design for solving Algebraic equations automatically.

History Maintenance of Given Equation The game will store the states of equation while solving. This will allow player to undo or redo his/her action.

Wining Perfect Score The game will provide an equation with right and wrong options. If a player selects every right option to solve equation, it will meet the perfect/ideal score i-e 100% score. This will creates motivation to get score close to ideal score in all levels.

Displaying Mistakes At the end of game, mistakes of applying wrong options, will also displayed with result. This will helps a player to learn from mistakes and play again with more knowledge.

3.3 Technical Description

The developed game is a mathematical game which allows player to solve mathematical equations by pressing provided options.

The flow of a game is executed by processes called modules. The game is based on several modules that are developed to make a game. The implementation of these modules are presented in Appendix A. A modular view of structure of a game is shown in Figure3.1. According to the figure, there are seven basic modules that are running behind the game:

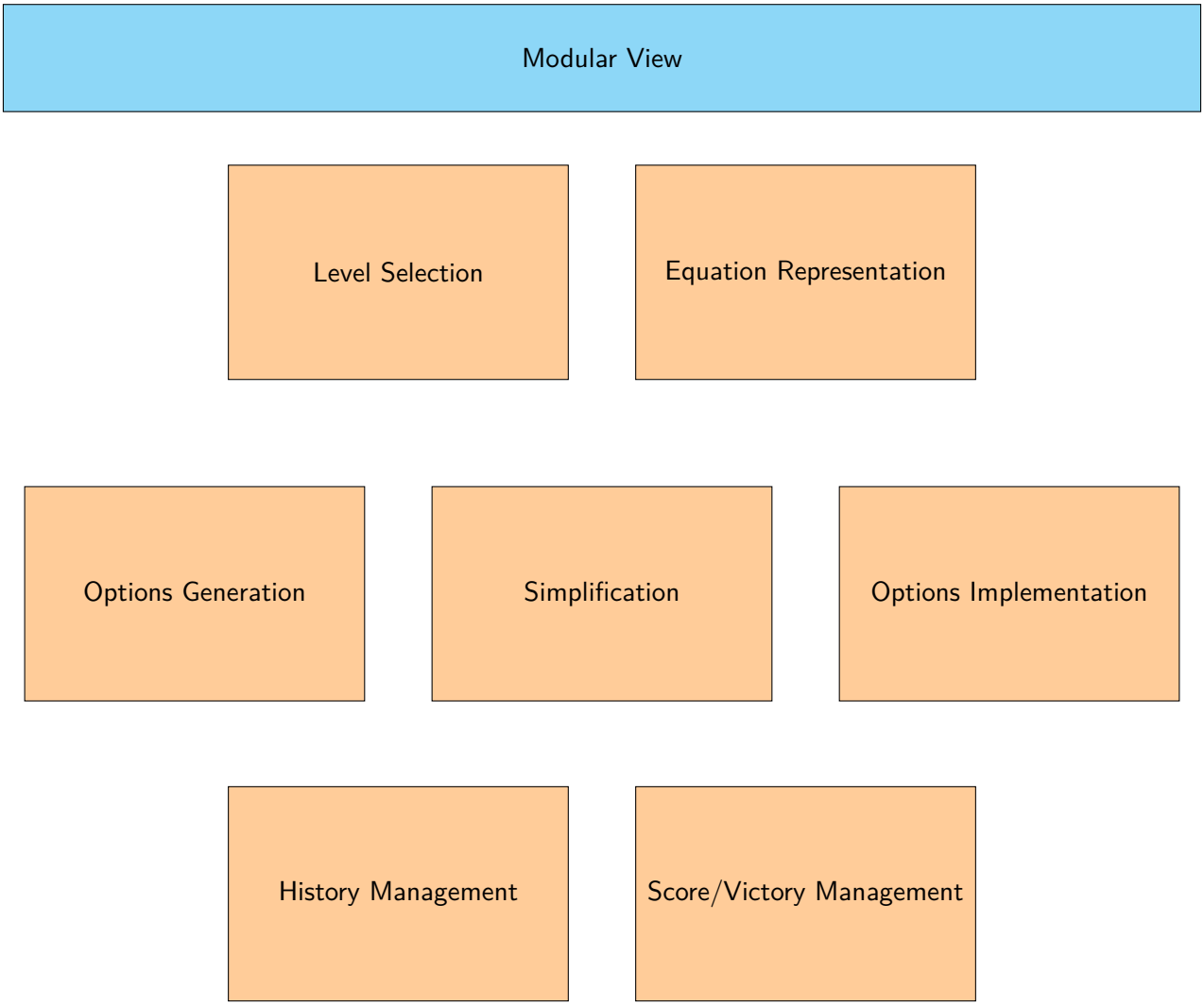

Figure 3.1: Structure of a Game By Modules. Figure describes seven basic modules that are running behind the game.

3.3.1 Level Selection

Level selection is an option given to the player to select the type of equation. It depends on student to practice any type of equation according to their comfort and requirement. There are three generic types of equation. Although its not actually considered as a type. It is a collection of three sets of equations that have different properties. i-e more addition, more division or complex multiplication is involved.

- **Easy: Basic Equations To Start:** It contains small and easy equations to make a player starts learning it with easiness.

- **Medium: Standard Equations To Practice:** It contains standard and regular equations to make a player practice the learning concept that gained in a classroom or in a basic level of game.
- **Hard: Complex Equations To Excel:** It contains slightly difficult and long equations in order to excel the concept and gain experience on balancing any equation.

Upon selection of level by a user, the level number is temporarily stored. Level number is a unique number given to every level. For basic:1, standard:2, complex:3. After storing level number, the equation is selected randomly from a selected level. It not only contains equation but also the related answer to compare afterwards. The Figure3.2 clearly explains the method of level selection.

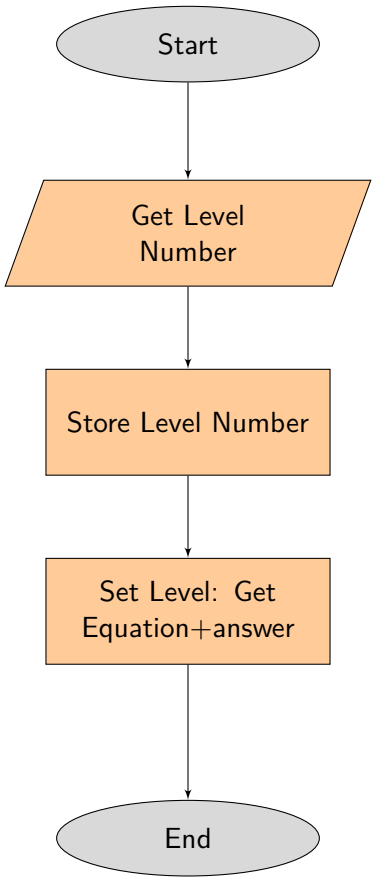

Figure 3.2: Flow Chart of Level Selection. The flow of selecting level and obtaining equation with related answer

3.3.2 Equation Representation

The equation is an expression and it is stored in the form of expression format. For example $2 + 3$ will be in the form of $sum(num(2), num(3))$, where num is a function giving information that 2 is a number and initialize it as NUM in order to use it for further processes like adding number or representing it in the form of LaTex etc. Below is the list of getting equation into information to process further:

- $NUM \Longrightarrow$ constant numbers (one expression) 2 : NUM(2)

- $VAR \Longrightarrow$ variable (one expression) x : VAR(x)
- $UMIN \Longrightarrow$ negative (one/two expression) -2 : UMIN(NUM(2))
- $UINV \Longrightarrow$ inverted (one/two expression) $1/2$: UINV(NUM(2))
- $SUM \Longrightarrow$ addition (two expression) 2+x : SUM(NUM(2),VAR(x))
- $SUB \Longrightarrow$ subtraction (two expression) 2-x : SUB(NUM(2),VAR(x))
- $MUL \Longrightarrow$ multiplication (two expression) 2x : MUL(NUM(2),VAR(x))
- $DIV \Longrightarrow$ division (two expression) 2/x : DIV(NUM(2),VAR(x))
- $EQ \Longrightarrow$ equating (two expression) $2+x = 3$: EQ(SUM(NUM(2),VAR(x)), NUM(3))

For every expression, there are sub expressions and every sub expression can produce further sub expression. For example $a = 2 + 3 + x$ can be represented as a = SUM(SUM(NUM(2),NUM(3)), VAR(x)). Then we can say sub expressions of a are b and c where b = SUM(NUM(2),NUM(3) and c = VAR(x). Note that the sub expression b can be further produce other sub expressions like NUM(2) and NUM(3). In this way, it will be easy to access every component in the equation to solve it.

Thus an equation is extracted in the form of expression format like $sum(num(2), num(3))$. Then this information is processed because it is a combination of several functions that initialize every component to its property as defined the list above. This will then translated to mathematical form easily and displayed in the game as shown in Figure3.3.

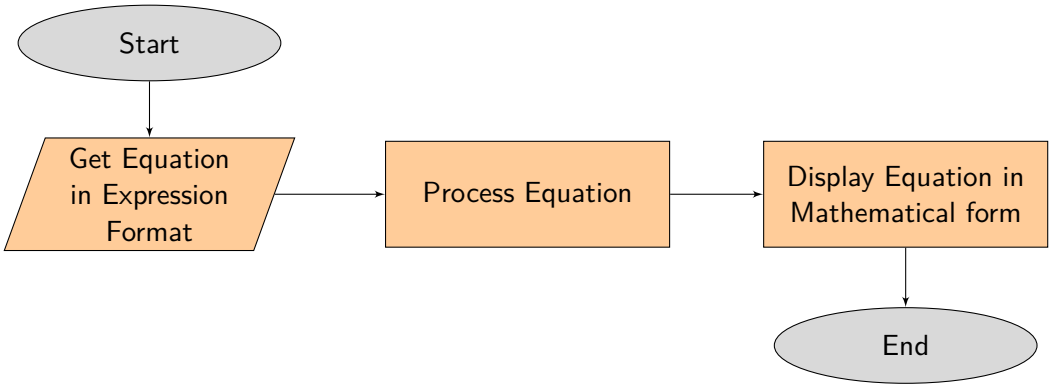

Figure 3.3: Flow Chart of Representing Equation in Mathematical Form

3.3.3 Option Generation

After the representation of equation, the related options according to equation has to be generated. Options are the available choices that a player can select to solve equation. There are both right and wrong options in the list. So it depends on player how to get a solution for equation. To generate options, the current equation has to be split by equality. As discussed above, the expression can produce sub expressions. Thus the given equation is processed to get sub expressions. There are two sub expressions. One is at right side of equality and other is at left side. For example $x + 2 = 3$ is split into $x + 2$ and 3. In actual, EQ(SUM(VAR(x),NUM(2)), NUM(3)) is converted to sub expressions SUM(VAR(x),NUM(2)) and NUM(3). This is done to generate options for both sides of equation. After that, previous options (if

present) are removed to stop overlapping with new ones. Next step will be to get and set options. While setting options, two things have to be set.

- **Action**: Add, Subtract, Divide or Multiply
- **Value**: Upon which action applied. i-e Subtract 2

For right options, right actions are set opposite to given expression. For example if it is $-x + 5$ then one option will be $subtract5$ and other will be $addx$. From right options, wrong options are created just by inversing the right actions. In this way, both type of options are obtained as shown in Figure3.4.

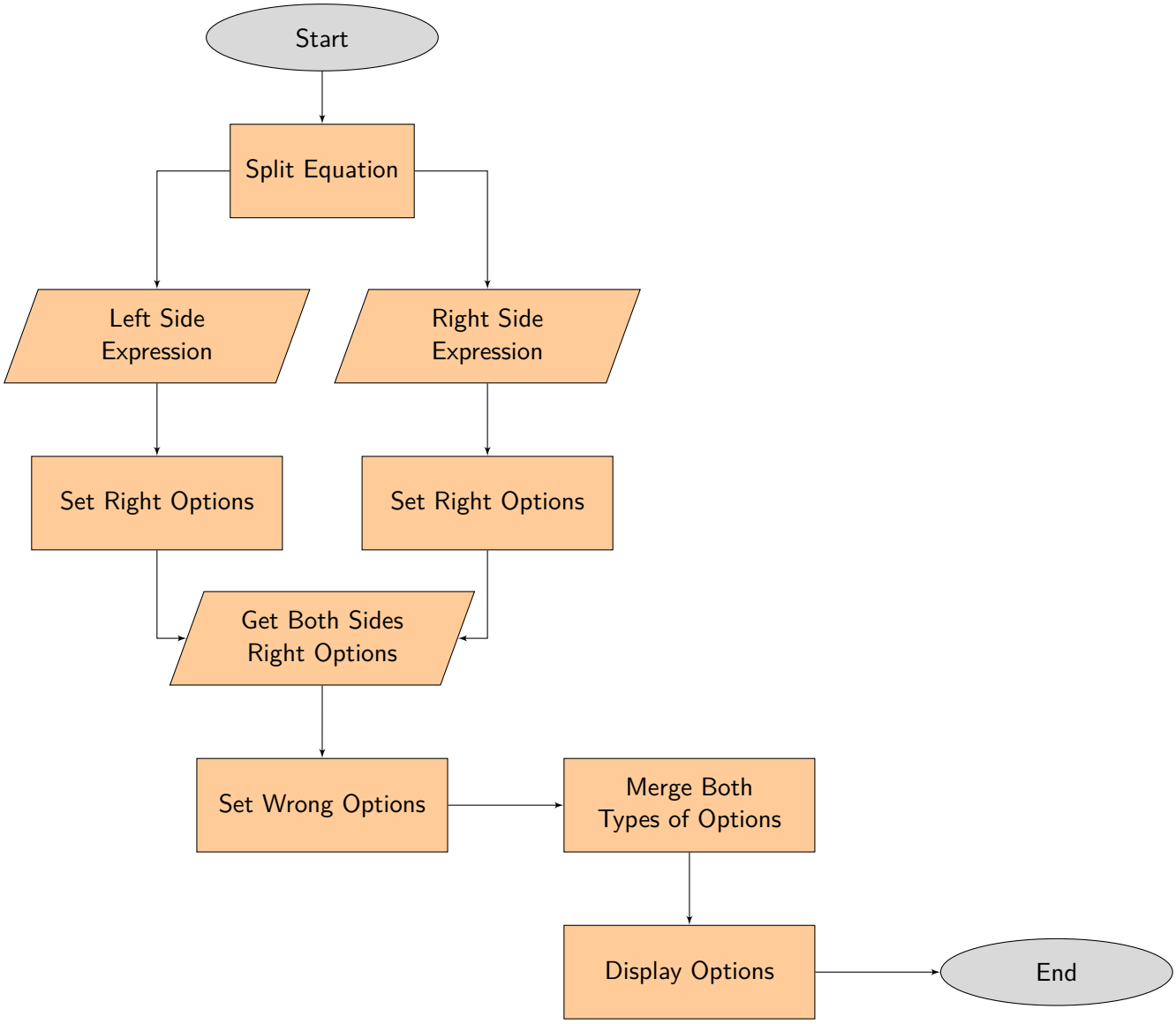

Figure 3.4: Flow Chart of Generating Options. Figure shows a method of generating right and wrong options according to the given equation.

3.3.4 Options Implementation

Option implementation is to apply the option on both sides of an equation in order to solve it. When a player selects any option (right or wrong), the action and value of that selected option is accessed. The action and value gives information about what to perform on both sides of equation. For example $add5$ will allow, to add 5 on both side of equation $x - 5 = 3$. Thus the equation becomes $x - 5 + 5 = 3 + 5$. After few seconds of time, it will simplify by the system in order to apply option i-e $x = 8$. This can be shown in Figure3.5.

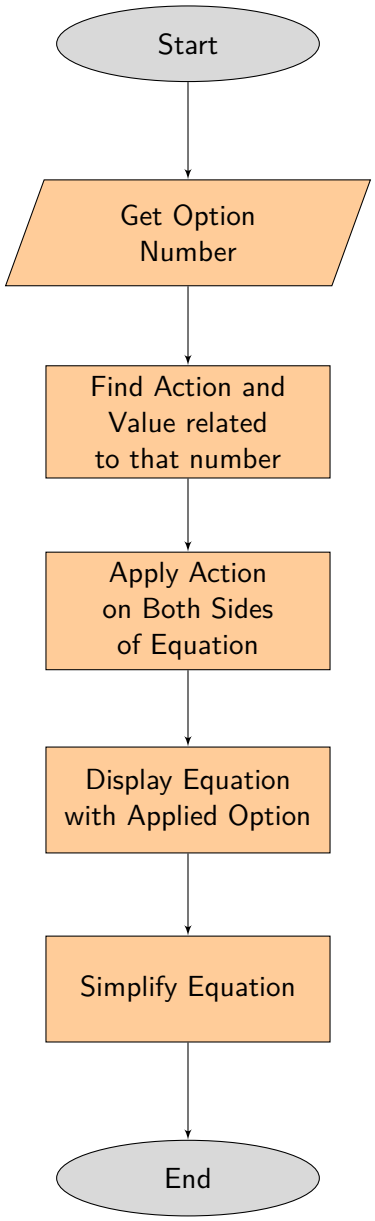

Figure 3.5: Flow Chart of Option Implementation on Equation. Figure shows a method of applying selected option on equation

3.3.5 Simplification

Simplification of equation is a core module of a game. Simplification is a process of calculating the similar terms in an equation and give them one value. In this way, the equation becomes shorter and leads to solution. Balancing 'X'is done by applying option and then simplify the equation. The equation is again split by equality into left and right sides. Then a process of simplification is applied. There are rules that simplify each side. Rules are made to identify the expressions and then solve accordingly. According to information of equation, there are four basic expressions i-e SUM, SUB, MUL and DIV. Thus there are four basic rule sets.

- Rules for SUM: Check the sub expressions of SUM and solve them.
- Rule for SUB: Convert to SUM and then check the sub expressions of converted SUM to solve it.
- Rules for MUL: Check the sub expressions of MUL and solve them.
- Rule for DIV: Convert to MUL and then check the sub expressions of converted MUL to solve it.

When an expression comes to the simplification, it will go through above mentioned rules. For rule of SUB, it will be converted into SUM. For example, taken $2-3x$ as $2+(-3x)$ or SUM(NUM(2),UMIN(VAR(3x))). Similarly for rule of div, it will be converted into MUL. For example, taken $2/3$ as $2*(1/3)$ or we can write it also like this: MUL(NUM(2),UNIV(3)). Where $UMIN$ and $UINV$ are helping functions to inverse or invert any term. Then apply SUM or MUL rules on it. In figure Figure3.6, simplification process is shown. While in Figure3.7 detailed function of simplify is explained.

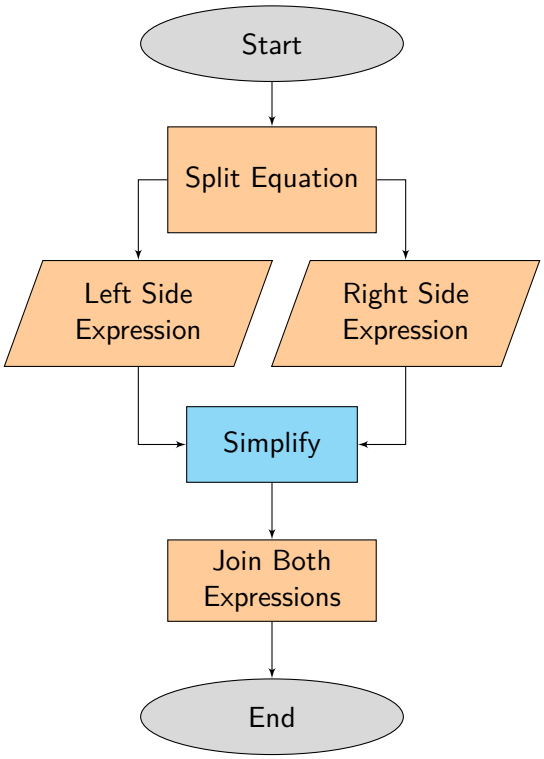

Figure 3.6: Flow Chart of Simplification Process of an Equation

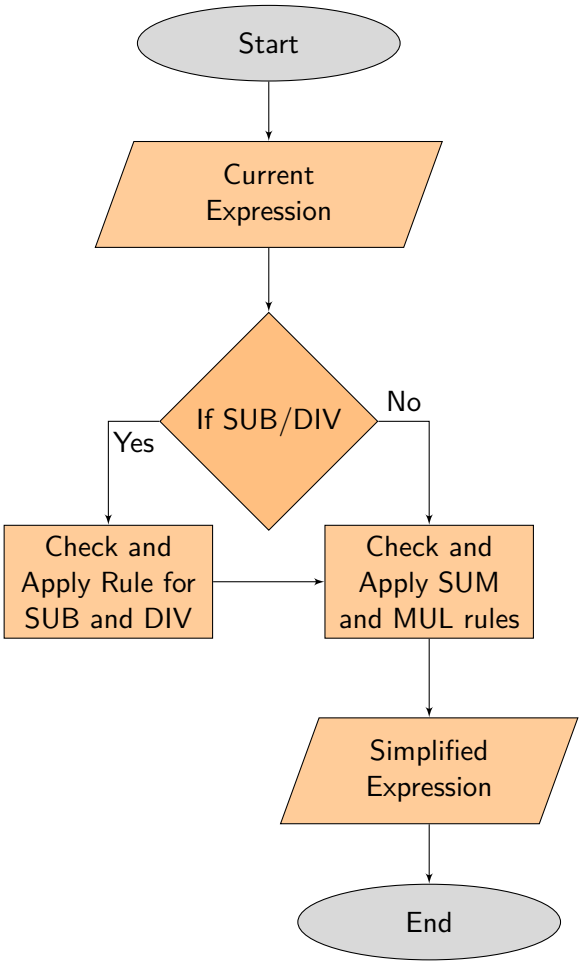

Figure 3.7: Flow Chart of Functionality of Simplify. Figure shows the detailed process of simplifying an equation.

The simplification process involve rules. Rule is a process of checking specific condition and then solve it accordingly. For example, if an expression is $2 + 3$, simplification process check which rule is applied and then apply the required rule to solve it into 5. Here, the addition of two numbers, is a sum rule of two numbers. Similarly, we have several combinations for making rules like sum of two numbers, two variables, two negative variables and so on. Thus making combinations for every property (SUM, MUL, SUB, DIV) will make complete rules. The main method to identify every combination is first recognize the variables and numbers. There are four types of variables i-e positive variable $PVAR$, negative variable $NVAR$, positive variable with coefficient $PVARC$, negative variable with coefficient $NVARC$. While two types of numbers i-e Positive number $PNUM$ and negative number $NNUM$. In an equation, expression contains sub expressions and these sub expressions can be a number or variable. Besides, these sub expressions can be another expression of SUM instead of any number or variable. For example, $SUM(2, 3)$ have two numeric sub expressions while $SUM(SUM(1, 2), 3)$ has one numeric sub expression but other is another expression that can evolve further two numeric subexpressions. Thus in addition to variables and numbers, sub expressions also contain another $SUM, MUL, SUBorDIV$. So for making combinations, all these factors have to be considered. There is following combinational method is given for each property. Figure3.8 shows all possible combinations of sub expressions of SUM. These combinations are the rules that will check the expression and provide solution accordingly. For example, lets consider combination-[1,2] as shown in the indicated figure. The combination-[1,2] or we can say $[PNUM, NNUM]$ is a rule of SUM.

This rule will check that first sub expression of SUM should be a positive number and second one should be a negative number. If this condition applies, then apply the solution and solve them.

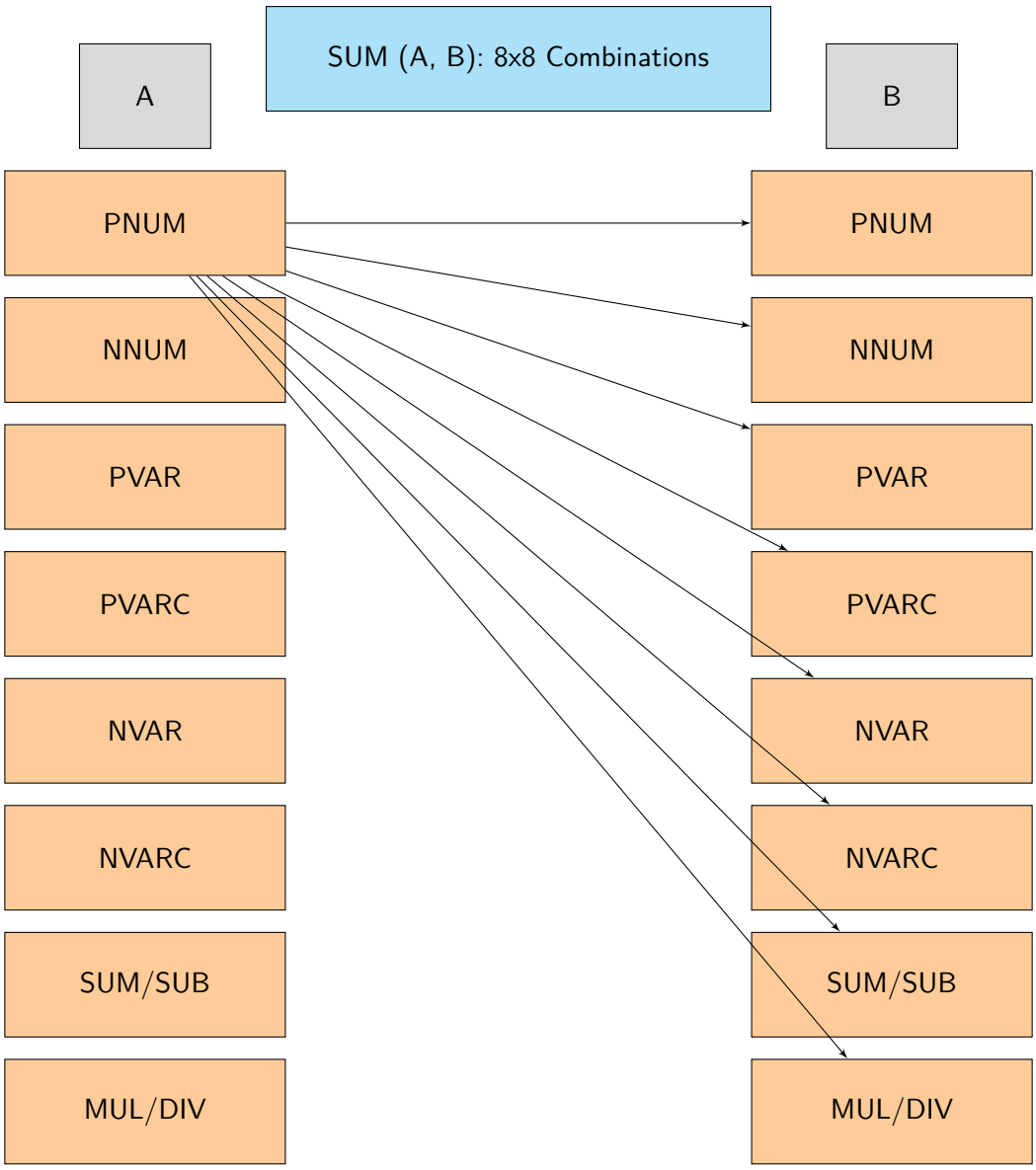

Figure 3.8: Simplification Rules for SUM. The figure shows all possible combinations of sub expressions A and B inside SUM.

Similarly, same rules combinations are applied for SUB but after passing through Rule for SUB. This rule convert expression of SUB into SUM as described above. Then the converted expression passed through the combinations for SUM-RULES. The Figure3.9 clearly describes the concept.

For expression of MUL, same method of combinations are applied as in SUM-Rules. In the Figure3.10, all possible combinations of sub expressions are defined.

Similarly, same MUL rules are applied for DIV but after passing through Rule for DIV. This rule convert expression of DIV into MUL as described above. Then the converted expression passed through the combinations for MUL-RULES. The Figure3.11 clearly describes the concept.

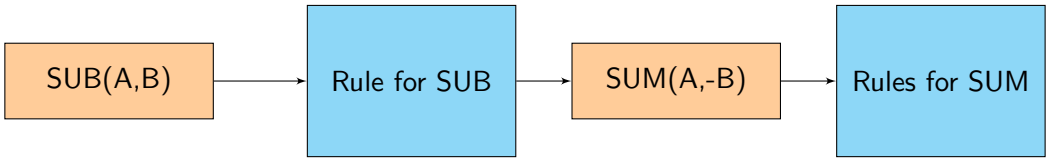

Figure 3.9: Rule for SUB. Conversion of expression into SUM and then passing to rules for SUM, where A and B are sub expressions inside SUB.

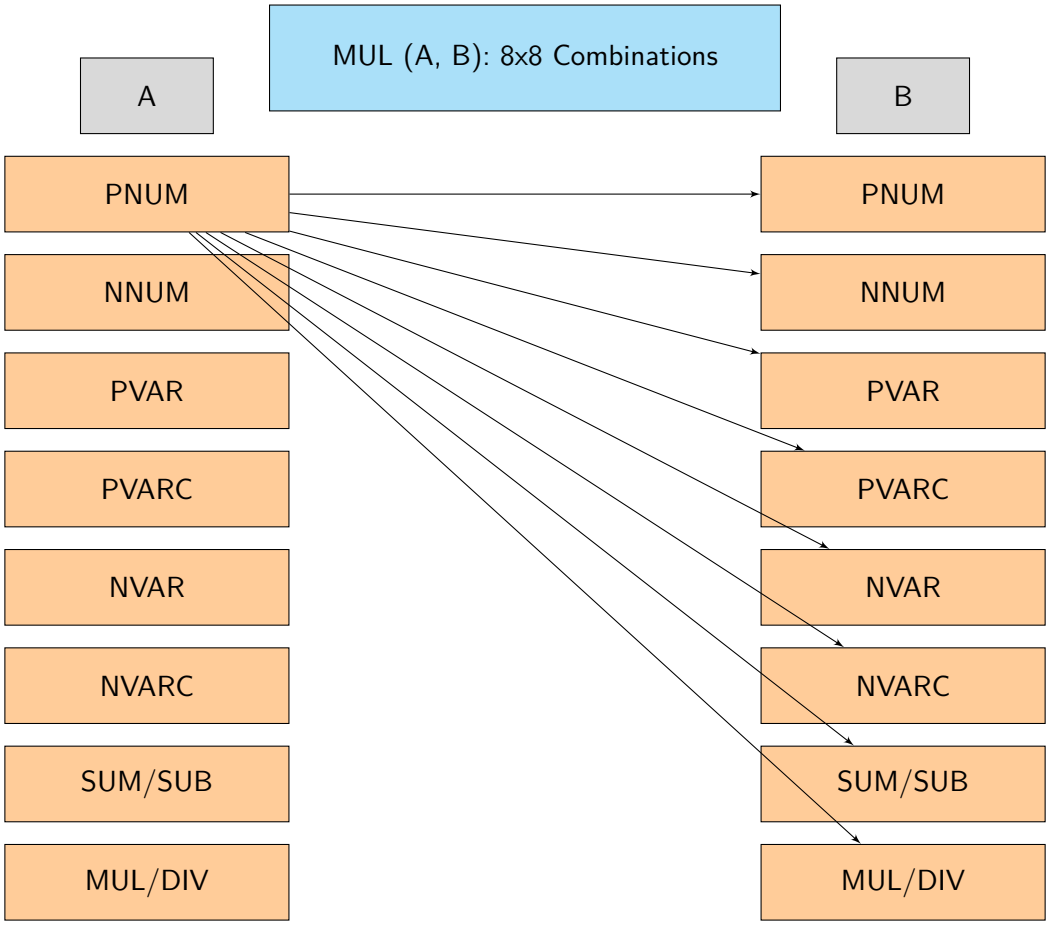

Figure 3.10: Simplification Rules for MUL. The figure shows all possible combinations of sub expressions A and B inside MUL.

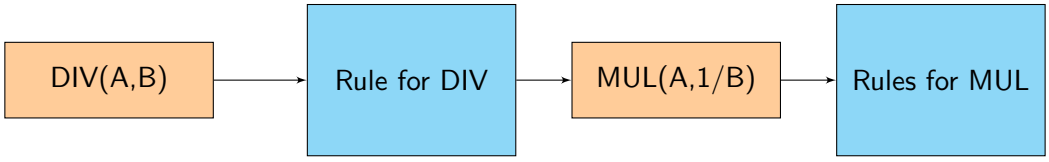

Figure 3.11: Rule for DIV. Conversion of expression into MUL and then passing to rules for MUL, where A and B are sub expressions inside DIV.

The equation is a combination of two expressions separated by a sign of equality. As described above, the equation is first split into right and left. These both expressions then go to the process of simplification. In this process, each expression is passed through the rules of SUM, SUB, DIV, MUL and check for appropriate rules to apply on expression. After applying rules, the simplified expressions are then again joined by equality to form equation but now its a simplified equation as shown in Figure3.12.

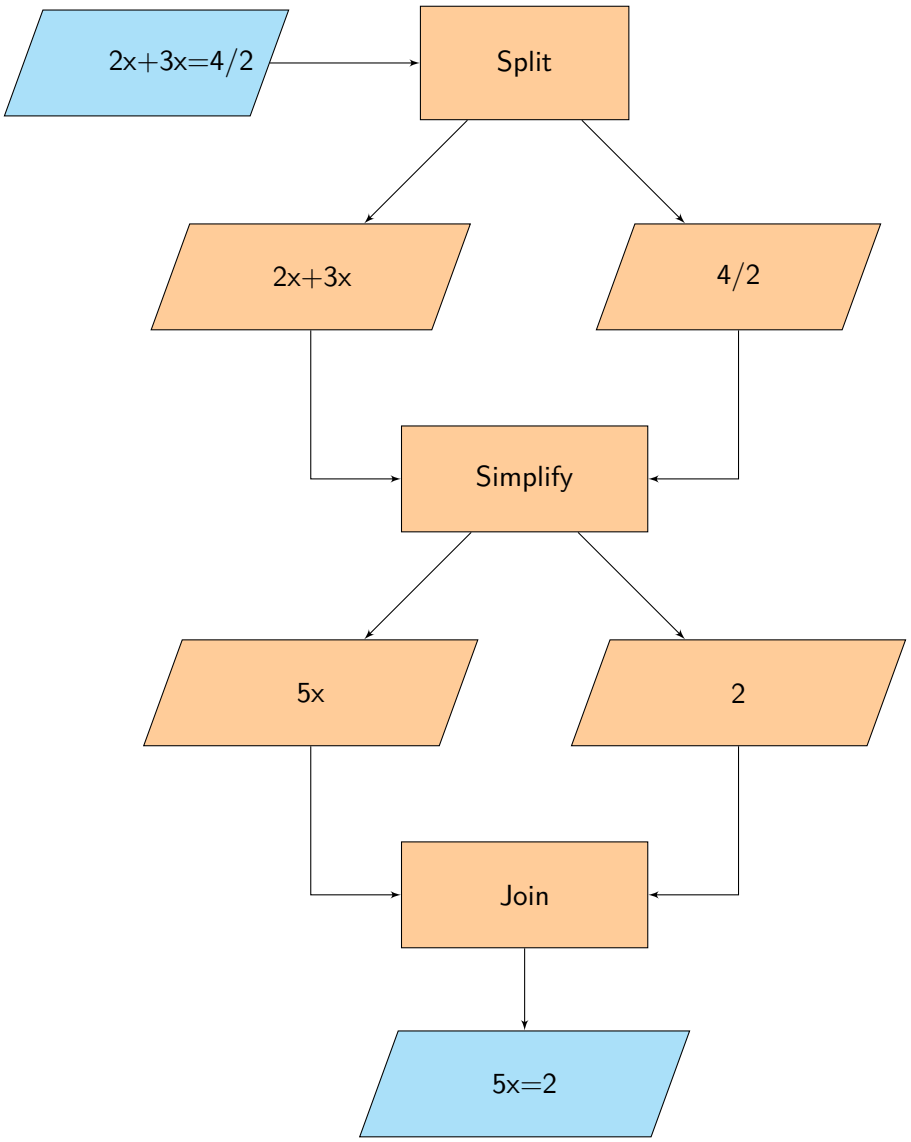

Figure 3.12: Example of Equation Passing Through Simplification Process.

3.3.6 History Management of Equation

The game is also capable to maintain the history of equation states. It allows a player to undo or redo his actions on the equation. For example, a player applied wrong option on the equation. At this point, a player is able to undo the applied option and goes to previous state of an equation. Similarly, if now at previous state, player feels that the last option was correct then he can redo the action and goes to the last state of equation.

The management of history is done by storing every state of an equation whenever an option is applied or any simplification is processed. Starting from initial or original equation, when an option is applied, the equation become changed. This will not lose the previous state. All the states will be stored for further need. After resolution of equation, these states will be cleared out. The state of equation is stored when option or simplification is applied as shown in Figure3.13.

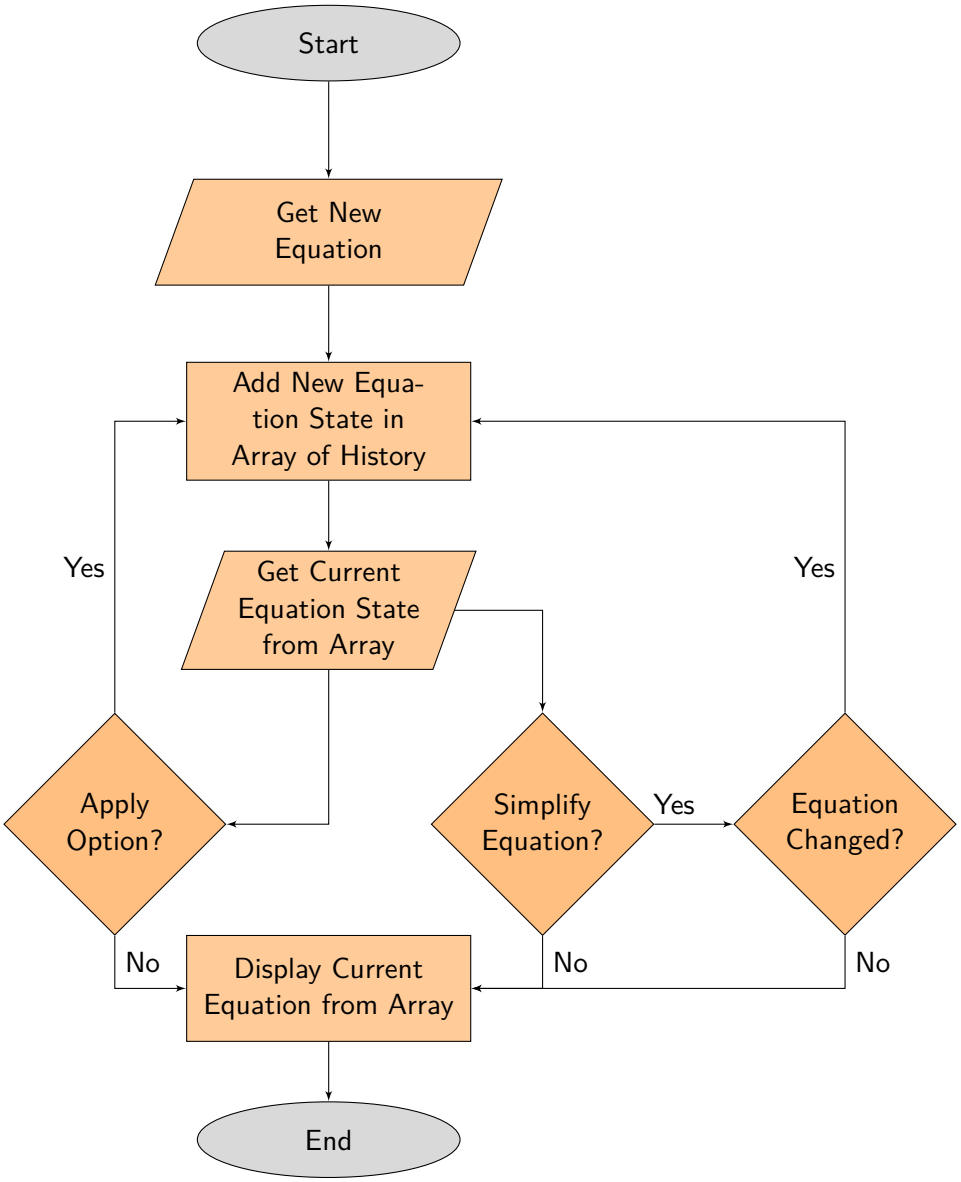

Figure 3.13: Flow Chart of History Management of Equation. The figure shows the flow of storing new equation whenever any option or simplification is applied on current equation.

In history management, a player can undo or redo the action. Undo process brings the previous state of equation before applied option or simplification. While Redo brings the last applied option or simplification.

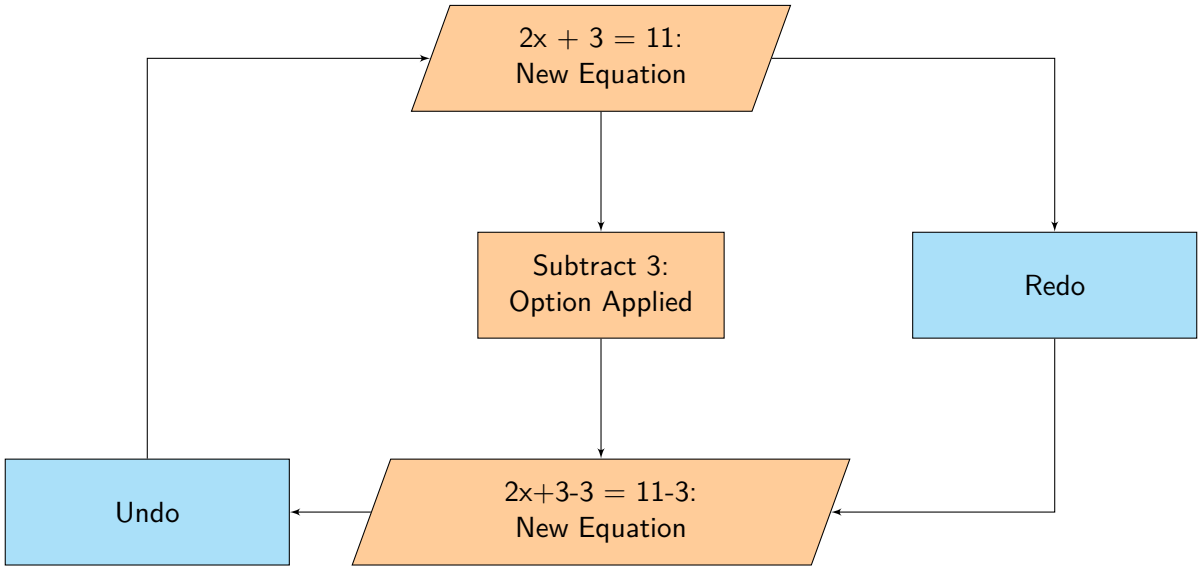

Figure 3.14: Basic Flow of Undo and Redo in an Equation. The figure shows an example to demonstrate the process of Undo and Redo.

3.3.7 Score Management

Upon successful resolution of equation, a player will be notified with the score. Score is like marks or grade as in school. The score is calculated upon the selection of option by a user. When a player selects right option, it will get positive score. While when a player selects wrong option, it will get negative score. The obtained score will be the sum of scores of pressing right options and wrong options. When an equation is loaded, the options related to it, also loaded. These options are the collection of right and wrong options. With every option, a score is associated. When a player selects that option, the associated score will also obtained with the option. This associated score then adds to current score of a player. The associated score or option score $OptScore$ is positive for right option and negative for wrong option.

$$Score = Score + OptScore$$

The percentage will also displayed after solving an equation. The score obtained and the ideal score is calculated to get percentage. Ideal score is the number of times a player presses an option and the option should be right. It is the ideal condition of a game that all the selected options are right. In short, ideal score is a sum of scores of pressing right options always. It is used to get percentage of a player.

$$Percentage = (Score \div IdealScore) * 100$$

Besides, a player can see where it made a mistake or selected wrong option. In this way, a player would be able to learn and improve further in order to obtain ideal score. Figure3.15 shows a flow of score management during solution of equation.

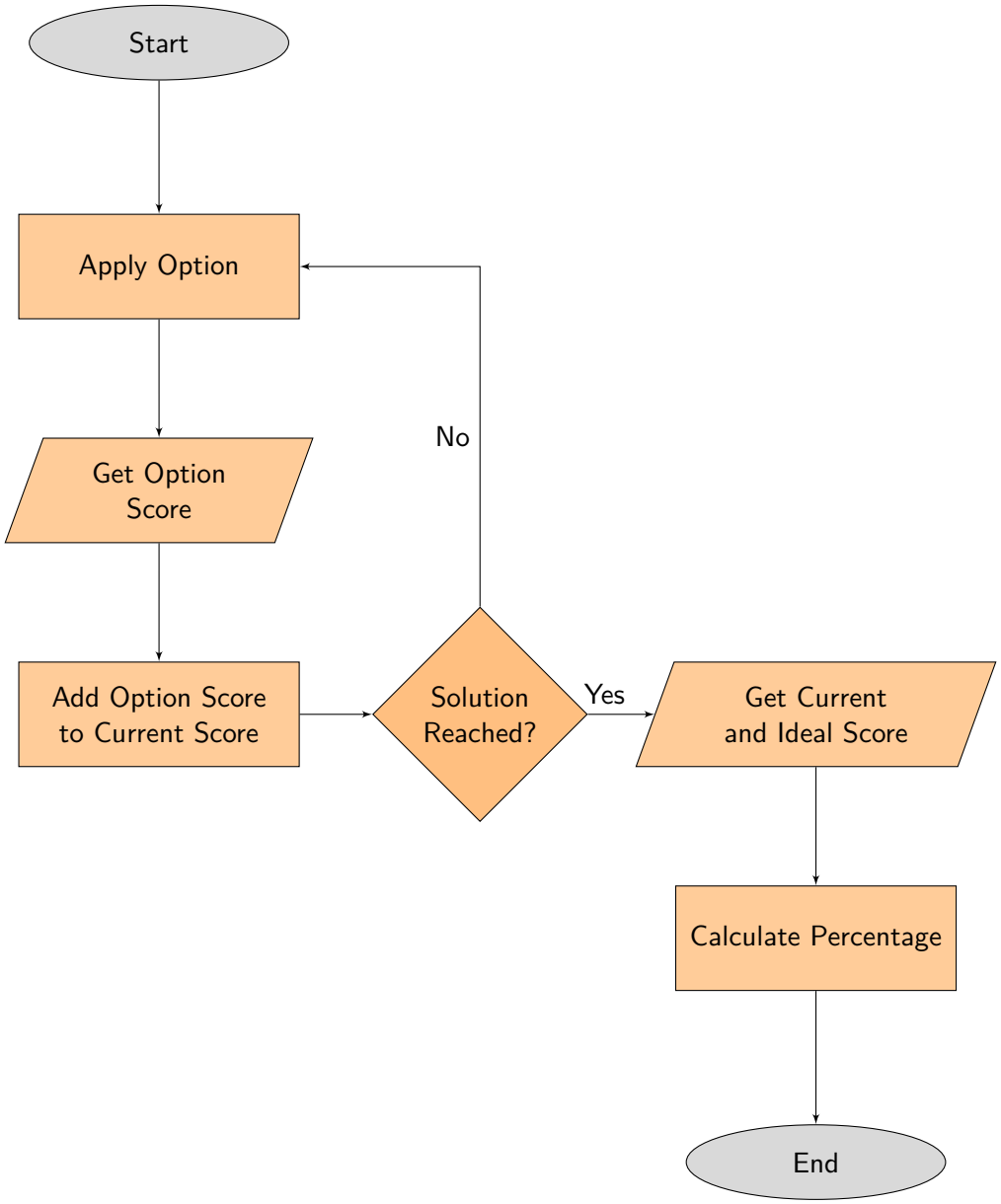

Figure 3.15: Basic Flow of Score Management. The figure shows calculation of score upon selection of option and then calculating percentage after reaching to solution of equation.

3.3.8 Main Flow of a Game

The main flow of a game is selecting level of equation, representing equation, loading options according to given equation, selecting option and then simplifying equation until the solution is achieved. The main flow according to user perspective is showed in Figure3.16. Note that the figure is not showing any winning or losing condition. It simply describes the basic flow of what the game is about. Similarly, in Figure3.17, main flow of a game is described according to system perspective.

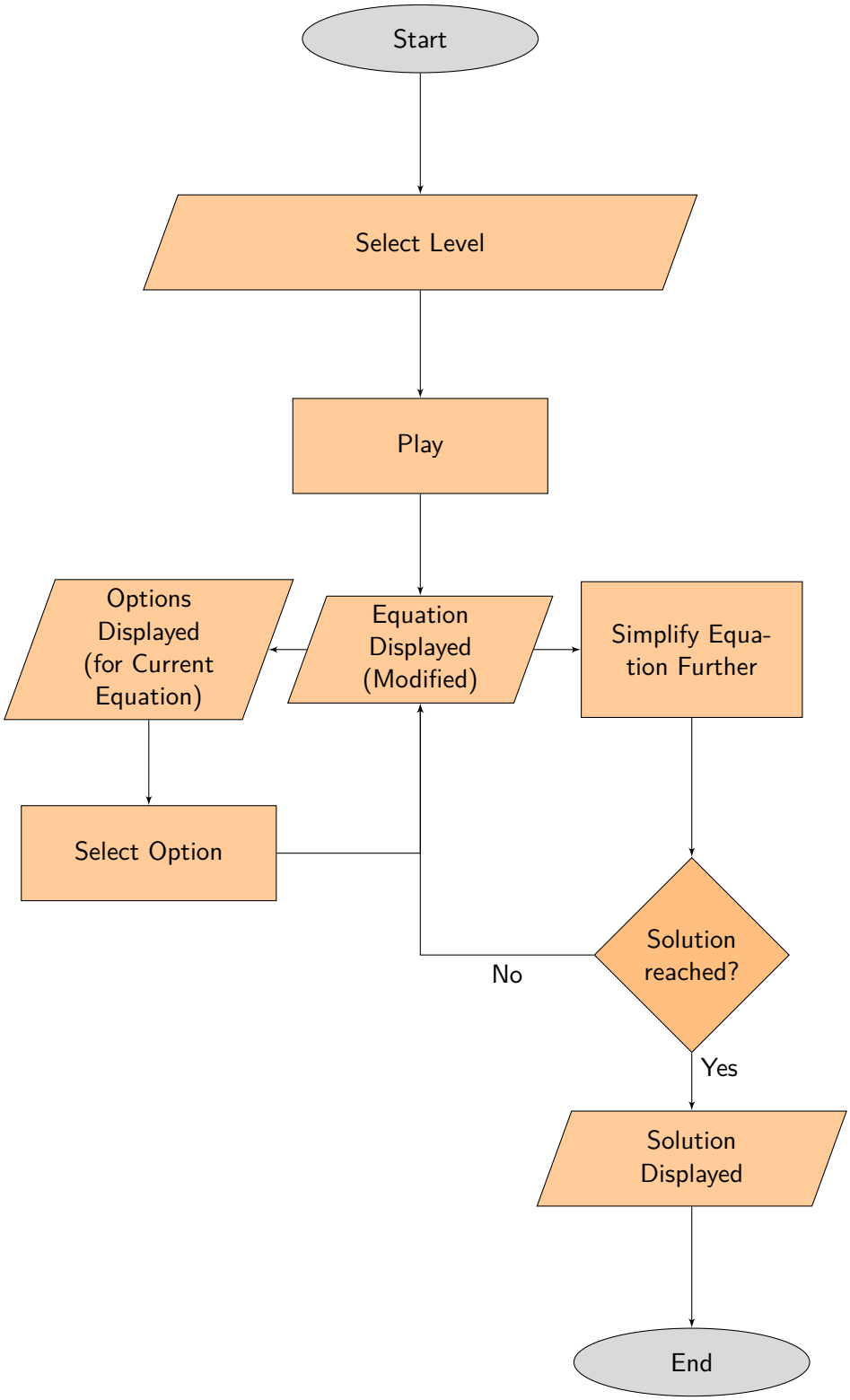

Figure 3.16: Main Flow of a Game from User's Perspective. Figure shows, how the game will proceed according to player perspective.

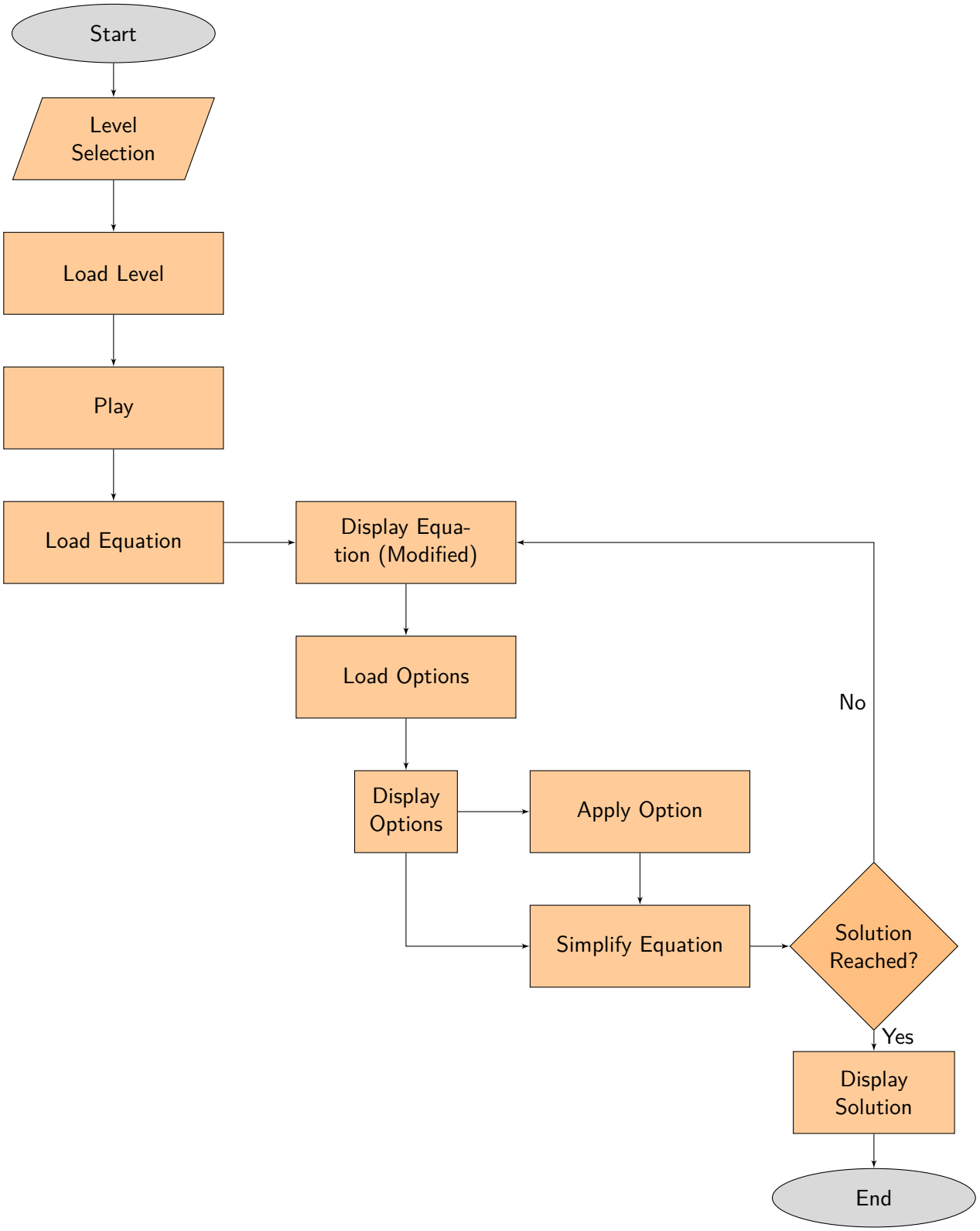

Figure 3.17: Main Flow of a Game from System Perspective. Figure shows, how the game will proceed according to system perspective.

3.4 Formal Description

The description of game in terms of mathematics is presented. The developed game have states, simplification rules, victory condition, actions, views and progression. Each of the factor is described below:

3.4.1 State

For start playing game or solving an equation, first thing is to get an equation from a level. The state defines the state of a player in a game with associated equation that needs to be solved.

- There are three levels $\{L_1, L_2, L_3\}$;
- Each level L_i have set of equations $\{EQ_1, EQ_2, EQ_3, ...EQ_n\}$ where $i = 1, 2, 3$;
- Each equation EQ_j of L_i where $j = 1, 2, ..n$ includes:
  - Set of equation states $EQ_j = \{e_0, e_1, e_2, ..e_n\}$;
  - Set of options for every equation state $O_k(e_k) = \{o_1, o_2, ...o_9\}$ where $k = 0, 1, 2..n$;.
- The target of equation solution is the last state of equation i-e $T_j = EQ_j(e_n)$.
- Initial state of game will be $S_0 = \{EQ_j(e_0, O_0), T_j\}$
- Current state will be $S_c = \{EQ_j(e_c, O_c), T_j\}$

3.4.2 Simplification Rules

Each question $\mathcal{Q}$ represents a mathematical equation EQ , containing a variable x that needs to be balance through simplification.

- Each EQ is a combination of two expressions $\mathcal{E}_1$ and $\mathcal{E}_2$ separated by an equality.

$$EQ \iff \mathcal{E}_1 = \mathcal{E}_2$$

- Each $\mathcal{E}$ is a combination of two sub-expressions A and B that are separated by Arithmetics Operators (OPR) i-e Addition (SUM), Subtract (SUB), Multiply (MUL) and Divide (DIV).

$$\mathcal{E} \iff \text{OPR}(A, B)$$

  where $\text{OPR} = \{\text{SUM}, \text{SUB}, \text{MUL}, \text{DIV}\}$
  - Expression for adding two sub expression: $\text{SUM}(A, B) \implies (A + B)$
  - Expression for subtracting two sub expression: $\text{SUB}(A, B) \implies (A - B)$
  - Expression for multiplying two sub expression: $\text{MUL}(A, B) \implies (A \times B)$
  - Expression for dividing two sub expression: $\text{DIV}(A, B) \implies (A \div B)$
- Each sub expression can produce another sub expressions or a term $\mathcal{T}$ that cannot be further breakdown into sub expression.
- Each term $\mathcal{T}$ can be a variable $\mathcal{V}$ or number $\mathcal{N}$ that cannot be further produce sub expressions.

- $\mathcal{N}$ has two types:
  - Positive Number (PNUM) $\longrightarrow n$
  - Negative Number (NNUM) $\longrightarrow -n$
- $\mathcal{V}$ has four types:
  - Positive Variable (PVAR) $\longrightarrow x$
  - Negative Variable (NVAR) $\longrightarrow -x$
  - Positive Variable with Coefficient (PVARC) $\longrightarrow ax$
  - Negative Variable with Coefficient (NVARC) $\longrightarrow -ax$
- Sub expression, variables and numbers are refer as Item (ITM) in a generalize form. ITM $= \{A, B, \mathcal{V}, \mathcal{N}\}$
- **If Expression is Sum:** Check and apply rules. There are total 8x8 combinations of rule as described above in technical description. Here twenty main rules are mentioned in the table 3.1 that will directly simplify the expression.
- **If Expression is Sub:** $A - B \longrightarrow applySumRule(A + (-B))$
- **If Expression is MUL:** Check and apply rules in the table 3.2. There are also 8x8 combinations but here main rules are mentioned that directly simplifies the sub expressions.
- **If Expression is Div:** $A \div B \longrightarrow applyMulRule(A \times (1/B))$

Table 3.1: SUM Simplification Rules. The table enlist all rules that will simplify sub expressions inside the expression of SUM. Where n_1, n_2 and n_3 are numbers. x is a variable and a, b, c are coefficients.

Name	Before		After	Name	Before		After
rs1	$n_1 + n_2$	$\Longrightarrow$	n_3	rs2	$x + x$	$\Longrightarrow$	$2x$
rs3	$-x + -x$	$\Longrightarrow$	$-2x$	rs4	$-x + x$	$\Longrightarrow$	0
rs5	$x + -x$	$\Longrightarrow$	0	rs6	$n_1 + -n_2$	$\Longrightarrow$	$-n_3$
rs7	$-n_1 + n_2$	$\Longrightarrow$	$-n_3$	rs8	$-n_1 + -n_2$	$\Longrightarrow$	n_3
rs9	$ax + bx$	$\Longrightarrow$	cx	rs10	$-ax + -bx$	$\Longrightarrow$	$-cx$
rs11	$-ax + bx$	$\Longrightarrow$	$(+/-)cx$	rs12	$ax + -bx$	$\Longrightarrow$	$(+/-)cx$
rs13	$x + bx$	$\Longrightarrow$	cx	rs14	$-x + -bx$	$\Longrightarrow$	$-cx$
rs15	$-x + bx$	$\Longrightarrow$	$(+/-)cx$	rs16	$x + -bx$	$\Longrightarrow$	$(+/-)cx$
rs17	$ax + x$	$\Longrightarrow$	cx	rs18	$-ax + -x$	$\Longrightarrow$	$-cx$
rs19	$-ax + x$	$\Longrightarrow$	$(+/-)cx$	rs20	$ax + -x$	$\Longrightarrow$	$(+/-)cx$

3.4.3 Victory Condition

The victory condition of a player is when a state of equation is equal to the target set of an equation or we can say, victory occurs when the state of an equation reached to the last state.

- Victory condition for each E_j of state $S = \{E_j(e, O), T_j\}$ is when $e = Tj$ or $e = e_n$ where e_n is the last state of equation.

Table 3.2: MUL Simplification Rules. The table enlist all rules that will simplify sub expressions inside the expression of MUL. Where n_1, n_2 and n_3 are numbers. x is a variable and a, b, c are coefficients.

Name	Before		After	Name	Before		After
rm1	$n_1 \times n_2$	$\Longrightarrow$	n_3	rm2	$\mathcal{V} \times \mathcal{V}$	$\Longrightarrow$	$\mathcal{V}^2$
rm3	$n_1 \times -n_2$	$\Longrightarrow$	$-n_3$	rm4	$-n_1 \times +n_2$	$\Longrightarrow$	$-n_3$
rm5	$-n_1 \times -n_2$	$\Longrightarrow$	n_3	rm6	$x \times n_2$	$\Longrightarrow$	n_2x
rm7	$x \times -n_2$	$\Longrightarrow$	$-n_2x$	rm8	$-x \times +n_2$	$\Longrightarrow$	$-n_2x$
rm9	$-x \times -n_2$	$\Longrightarrow$	n_2x	rm10	$ax \times n_2$	$\Longrightarrow$	n_2ax
rm11	$ax \times -n_2$	$\Longrightarrow$	$-n_2ax$	rm12	$-ax \times n_2$	$\Longrightarrow$	$-n_2ax$
rm13	$-ax \times -n_2$	$\Longrightarrow$	n_2ax	rm14	$n_1 \times x$	$\Longrightarrow$	n_1x
rm15	$-n_1 \times x$	$\Longrightarrow$	$-n_1x$	rm16	$n_1 \times -x$	$\Longrightarrow$	$-n_1x$
rm17	$-n_1 \times -x$	$\Longrightarrow$	n_1x	rm18	$n_1 \times bx$	$\Longrightarrow$	n_1bx
rm19	$-n_1 \times bx$	$\Longrightarrow$	$-n_1bx$	rm20	$n_1 \times -bx$	$\Longrightarrow$	$-n_1bx$
rm21	$-n_1 \times -bx$	$\Longrightarrow$	n_1bx	rm22	$n_1 \times 1/n_2$	$\Longrightarrow$	n_3
rm23	$n_1 \times 1/-n_2$	$\Longrightarrow$	$-n_3$	rm24	$-n_1 \times 1/n_2$	$\Longrightarrow$	$-n_3$
rm25	$-n_1 \times 1/-n_2$	$\Longrightarrow$	n_3	rm26	$x \times 1/x$	$\Longrightarrow$	1
rm27	$x \times 1/-x$	$\Longrightarrow$	-1	rm28	$x \times 1/-x$	$\Longrightarrow$	-1
rm29	$-x \times 1/-x$	$\Longrightarrow$	1	rm30	$ax \times 1/bx$	$\Longrightarrow$	n_3
rm31	$-ax \times 1/-bx$	$\Longrightarrow$	n_3	rm32	$-ax \times 1/bx$	$\Longrightarrow$	$-n_3$
rm33	$ax \times 1/-bx$	$\Longrightarrow$	$-n_3$	rm34	$x \times 1/bx$	$\Longrightarrow$	n_3
rm35	$-x \times 1/-bx$	$\Longrightarrow$	n_3	rm36	$-x \times 1/bx$	$\Longrightarrow$	$-n_3$
rm37	$x \times 1/-bx$	$\Longrightarrow$	$-n_3$	rm38	$ax \times 1/x$	$\Longrightarrow$	a
rm39	$-ax \times 1/-x$	$\Longrightarrow$	a	rm40	$-ax \times 1/x$	$\Longrightarrow$	$-a$
rm41	$ax \times 1/-x$	$\Longrightarrow$	$-a$	rm42	$x \times 1/n_2$	$\Longrightarrow$	n_3x
rm43	$x \times 1/-n_2$	$\Longrightarrow$	$-n_3x$	rm44	$-x \times 1/n_2$	$\Longrightarrow$	$-n_3x$
rm45	$-x \times 1/-n_2$	$\Longrightarrow$	n_3x	rm46	$ax \times 1/n_2$	$\Longrightarrow$	n_3x
rm47	$ax \times 1/-n_2$	$\Longrightarrow$	$-n_3x$	rm48	$-ax \times 1/n_2$	$\Longrightarrow$	$-n_3x$
rm49	$-ax \times 1/-n_2$	$\Longrightarrow$	n_3x	rm50	$n_1 \times 1/x$	$\Longrightarrow$	$n_1 \div x$
rm51	$-n_1 \times 1/x$	$\Longrightarrow$	$-n_1 \div x$	rm52	$n_1 \times 1/-x$	$\Longrightarrow$	$n_1 \div -x$
rm53	$-n_1 \times 1/-x$	$\Longrightarrow$	$n_1 \div x$	rm54	$n_1 \times 1/bx$	$\Longrightarrow$	$n_1 \div bx$
rm55	$-n_1 \times 1/bx$	$\Longrightarrow$	$-n_1 \div bx$	rm56	$n_1 \times 1/-bx$	$\Longrightarrow$	$n_1 \div -bx$
rm57	$-n_1 \times 1/-bx$	$\Longrightarrow$	$n_1 \div bx$				

3.4.4 Player's Action

The action of a player is to select the one option from different options of an equation state i-e $O_i(e_i) = \{o_1, o_2, ...o_9\}$. Thus the action A of player P will be:

$$A_i(P, o_k),\ iffO_i(e_i)\ where\ k = 1, 2, ..9$$

.

3.4.5 Evolution/Progression

The evolution of a game is how the game will progress. It starts from initial state. Then a player applied option called action and the state will change. Similarly, actions change the states and the game continues.

- The initial state of a player:
$$S_0(P) = \{E_j(e_0, O_0), T_j\}$$
- When action applied:
$$S_0(P) + A_0(P, o_n) \Longrightarrow S_1(P)$$
- Progress:
$$S_i(P) + A_i(P, o_k) \Longrightarrow S_n(P)$$

3.4.6 Player's view

- In a single player's view, current equation, options and score are involved. The player keeps on seeing current score SC after applying an option. Thus the view of a player is state and current score.
$$V(P) = S(P) + SC(P)$$

3.5 Game Testing

Game testing includes the testing of every module (as described in technical description) that makes a running game. There are seven modules that needs to be tested. For every module, there might be more than one processes that go for testing. Thus we can say, for every module, there might be more than one test cases.

3.5.1 Testing of Selecting Level

When a player selects level of a game, it is stored in a game to select equation from that level afterwards.

- **Test Case Name:** Verifying the working of level selection.
- **Objective:** To check the functionality of selecting level.
- **Conditions:**
  - The player should selects level.
  - The system interprets level in the form of number.
  - The system should stores the level number.

Table 3.3: Testing Outcomes for Level Selection

Input: Level Selected	Expected Output: Level Stored	Actual Output: Level Stored
Easy	1	1
Medium	2	2
Hard	3	3

3.5.2 Testing of Choosing Equation

When a player selects level of a game, it is stored in a game. Then the equation is selected randomly from that level.

- **Test Case Name:** Verifying the working of equation selection.
- **Objective:** To check the functionality of selecting equation.
- **Conditions:**
  - The system should have level number stored.
  - The system should matched the number with levels.
  - If level is 1, then selects equation from basic level examples.
  - If level is 2, then selects equation from standard level examples.
  - If level is 3, then selects equation from complex level examples.

Table 3.4: Testing Outcomes for Equation Selection

Input (level)	Expected Output	Actual Output
1: (level = Basic)	[equation] from Basic Examples	Basic (equation)
2: (level = Standard)	[equation] from Standard Examples	Standard (equation)
3: (level = Complex)	[equation] from Complex Examples	Complex (equation)

3.5.3 Testing of Representing Equation in Mathematical Form

When an equation is selected, it is processed in mathematical form and displayed to the user. The equation presented to the user in mathematical form and the equation that will process further for solution, should be the same. Besides, the equation should also correctly converted to mathematical form.

- **Test Case Name:** Verifying the conversion of equation in mathematical form.
- **Objective:** To check that equation selected and equation displayed in mathematical form, are equal.
- **Conditions:**
  - The system should get the equation in the form of information.
  - The system process the information and convert it into mathematical form.
  - Represent mathematical form of equation in a game.
  - The represented equation and the equation that will process further should be the same.

Table 3.5: Testing Outcomes for Equation Representation

Input (Equation)	Expected Output	Actual Output
EQ (SUM (UMIN (VAR ('x')), UMIN (NUM (5))), SUB (NUM (7), MUL (NUM (8), VAR ('x')))	-x -5 = 7 - 8x	-x - 5 = 7 - 8x

3.5.4 Testing of Option Generation

When an equation is selected and presented, the right and wrong options related to it should also be generated and displayed.

- **Test Case Name:** Verifying the working of option generation module.
- **Objective:** To check that both right and wrong options generated related to expressions present in an equation.
- **Conditions:**
  - The system should get the equation in the form of sub expressions by splitting the equation.
  - Every sub expression is processed to generate right options.
  - From right options, wrong options should be generated.

Table 3.6: Testing Outcomes for Option Generation

Input	Expected Output	Actual Output
NUM(2)	Right Option: Subtract 2 Wrong Option: Add 2	Right Option: Subtract 2 Wrong Option: Add 2
MUL(NUM(2),VAR(x))	Right Option: Divide 2 Wrong Option: Multiply 2	Right Option: Divide 2 Wrong Option: Multiply 2

3.5.5 Testing of Option Implementation

The equation is presented and the right/wrong options related to it are displayed in a game. When a player selects any option, the equation displayed, changes itself according to the selected option.

- **Test Case Name:** Verifying the working of option implementation module.
- **Objective:** To check that the chosen option is applied on equation.
- **Conditions:**
  - The user should selects option.
  - The system should take action and value information from selected option.
  - The action should applied on current equation with related value.

Table 3.7: Testing Outcomes for Option Implementation

Input	Expected Output (Equation)	Actual Output (Equation)
Option: Add 5 Equation: -x - 5 = -8x + 7	-x - 5 + 5 = -8x + 7 + 5	-x - 5 + 5 = -8x + 7 + 5
Option: Divide 10 Equation: 10x = 7	10x / 10 = 7 / 10	10x / 10 = 7 / 10

3.5.6 Testing of Simplification Rules

When an option is applied, next step is to solve equation. The main core of a game is in this module. Simplification is done by rules. Rules are a small processes of checking specific condition and then solve accordingly. Testing of rule leads to testing of simplification.

- **Test Case Name:** Verifying the working of rules in a simplification process.
- **Objective:** To check that the rules are simplifying the equation.
- **Conditions:**
  - The system should simplify the equation
  - The equation should be split into expressions.
  - The expressions should be passed through the set of rules.
  - If any rule is matched, apply the rule on that expression.
  - Join the simplified/non-simplified expressions to form equation again.

Table 3.8: Testing Outcomes for Simplification Rules

Input	Expected Output	Actual Output
SUM (NUM(5), NUM(5))	NUM(10)	NUM(10)
MUL (NUM(0), VAR('x'))	NUM(0)	NUM(0)
MUL (NUM(1), VAR('x'))	VAR(x)	VAR(x)
SUB (VAR('x'), NUM(6))	SUM (VAR(x), UMIN (NUM(6)))	SUM (VAR(x), UMIN (NUM(6)))
SUB (NUM(2), NUM(6))	UMIN (NUM(4))	UMIN (NUM(4))
SUM (UMIN (VAR ('x')), UMIN (VAR ('x')))	UMIN (MUL (NUM(2), VAR('x')))	UMIN (MUL (NUM(2), VAR('x')))
DIV (NUM(6), NUM(2))	NUM(3)	NUM(3)

3.5.7 Testing of History Management

The states of equation are preserved so that according to the need, a player can access it. Just like to do example on a paper, student can go back to previous equation. Besides, redo option is also there if a player wants to come back again to the last state. It should be needed that all the states are stored and working.

- **Test Case Name:** Verifying the working of history management module.
- **Objective:** To check that a player can undo and redo the states of equation.
- **Conditions:**
  - The states of equations should be stored.
  - When a player undo the equation, a system should get and display the previous equation.
  - When a player redo the equation, a system should get and display the last (recent) equation again.

Table 3.9: Testing Outcomes for Undo and Redo Operations

Input	Expected Output	Actual Output
Undo	Equation[Given - 1]	Equation[Given - 1]
Redo	Equation[Given + 1]	Equation[Given + 1]

3.5.8 Testing of Scoring System

When an option is applied, the score related to that option is added to current score of a player. If an option is right, positive score added while for wrong option, negative score added. The score of a player should be the sum of scores of right and wrong options. For right option, five points are added to current score while for wrong option, deduct 1 from current score.

- **Test Case Name:** Verifying the working of score management module.
- **Objective:** To check that the sum of scores of both right and wrong options are added to current score.
- **Input Specifications:**
  - Data: Option Applied that gives Selected Option Score.
  - Type: Numeric
  - Range: -$n, ., 0, ., +n$
- **Conditions:**
  - The user should selects option.
  - The system should take option score from selected option.
  - Add score of related option only if an option is applied on an equation.
  - If option is right, add positive score.
  - If option is wrong, add negative score.
  - Calculate the sum of all positive scores and negative scores to get total score.

Table 3.10: Testing Outcomes for Score Management

Input	Expected Output	Actual Output
5 Right Option	Score = Score + 5	Score = Score + 5
1 Wrong Option	Score = Score + (-1)	Score = Score + (-1)

Table 3.11: Testing Outcomes of Total Score Management

Input (Selected Options)	Expected Output (Score)	Actual Output (Score)
r,w,r,r,w,r r = Right Option = 4 w = Wrong Option = 2	18 (4 x 5 + 2 x (-1))	18
rrrr r = Right Option = 4 w = Wrong Option = 0	20 (4 x 5 + 0 x (-1))	20

3.6 Play Testing

Play testing is a very vital step in game development. Successful game is not only about good mechanics, actions or rewards. Its about interest of a player to play it again.

3.6.1 Goals Behind Play Testing

There are several goals behind play testing of 'X in Balance' in order to make it useful and interesting for students.

- To get the response of students before making it a complete version. After all, it is a game for students of mathematics. Thus it is very important to get feedback from the students.
- To check the interest level of playing again. It is important both from game view and practicing view. If a player do not want to play again, then it could not practice more equations. Hence it will then leads to failure of a game.
- To know the complexity and easiness of a game by a real audience. Too much easy and complex game leads to boredom.
- To check the entertainment factor. The fun level achieved by game. Since it is an edutainment game, so entertainment is also running parallel to learning.
- To investigate the main objective that either a student learned from it or not.

3.6.2 Proposed Method for Play Testing

There is a framework developed for testing the game play. For this, a group of specific age students are needed to play game on their smart phones or tablets. After playing the game, a questionnaire will be given to them. There are several parameters that are present in questionnaire. Each parameter would be evaluated by the students. Additionally, a test will be taken on a paper that either a student learned something or not. The process of play testing includes following items and functions:

- **Audience for Play Testing:** School going students.
- **Age Group of Audience:** min: 9, max: 12.
- **Audience Class/Grade:** min: 4th, max: 6th.
- **Number of Audience Participate:** min: 10.
- **Things Needed to Play:** Smart phone/Tablet, Internet Access, Course Book for guidance.
- **Parameters to Evaluate by Audience after Playing Game:**
  - **Comfort Level:** It evaluates the design of a game. The rules, functionality and setup of a game should not be too complex, to make effort to first understand them instead of playing the game.
    * **Statement:** Exploring Rules and Functionality of a Game.
    * **Range:** {0,1,2} where 0 means could not understand the game rules, 1 means, it took some time and 2 means, it easily explored.

Table 3.12: Table for Receiving Feedback about Comfort Level

Levels	0 Could not understand the game rules	1 Took some time to explore	2 Easily explored the rules
Comfort Level			

  - **Difficulty Level:** It includes the difficulty level that a player may face while solving the equation. It should not be too much easy that a player gets bored. Besides, it should not be too hard that a player distract from a game.
    * **Statement:** Difficulty While Solving Equation.
    * **Range:** {Easy, Medium, Hard}.

Table 3.13: Table for Receiving Feedback about Difficulty Level

Levels	Easy	Medium	Hard
Difficulty Level (Solving Equation)			

  - **Fun Level:** The entertainment factor provided by game. It should be achieved high so that a player's interest level could not be break.
    * **Statement:** How Much You Enjoyed? Fun Level.

Table 3.14: Table for Receiving Feedback about Fun Level

Levels	Very Bad	Bad	Normal	Good	Very Good
Fun Level)					

  * **Range:** {Very Bad, Bad, Normal, Good, Very Good}.

– **Want to Play More? :** This question itself, gives half of the feedback from user. The main focus of game is learning and it can only be achieved, when a player practice more equations. For this, a player should have to play more. Thus, "play more" leads to "learn more". This automatically then, leads to success of an educational game.
  * **Statement:** Want to Play Again in Future.
  * **Range:** {Yes, No}.

Table 3.15: Table for Receiving Feedback about Play More

Question	Yes	No
Want to Play More in Future?		

- **Results Compilation:** Get the feedback from all students and plot graph for every parameter, to check and analyze final result.
- **Improvements:** After analyzing result with expected outcomes, improvements will be made to enhance the game.
- **Random Course Questions:** It can be asked after playing that game in order to know either the student learned or not.

3.6.3 Structure of a Prototype for Play Testing

The structure of prototype is presented here. It is described in order to present the basic game play, that the students will experience during play testing. A detailed demonstration of prototype is mentioned in Appendix B. Following is the sequence of game play:

- **Starting Page:** Starting page of the game shows several options. Choose the option to play a game as shown in Table 3.16. This will leads to level selection.

Table 3.16: Starting Page and their Options

Starting Page of a Game	
Options	Play a Game
	How to Play
	Exit

- **Level Selection Page:** Choose the level and proceeds to solving of equation as shown in Table 3.17. Easy level is a basic level to starts exploring concept. Medium level is to practicing equations while hard level is for mastering the complex equations.

Table 3.17: Level Selection Page with Options

Level Selection Page	
Options	Easy
	Medium
	Hard
	Main Menu

- **Equation Solving Page:** After choosing level, an equation is displayed from that level to solve. With the equation, a set of right and wrong options (with dummy options also) are displayed as shown in Table 3.18. Both the options are shuffled. There are total nine options displayed on a screen. The player is able to select any of the options. Then the system will apply that option. Besides, there are three buttons that will do the working of undo, redo and simplify, if necessary.

Table 3.18: Equation Solving Page and its Functionalities

Solving Equation Page		
Equation	Options	Right Options
		Wrong Options
		Dummy Options
	Simplify	
	Undo	
	Redo	

  - **Applying Option in Equation Solving Page:** The selection of option button in a given page, changes the state of equation as shown in Table 3.19. Adding options to current equation will leads to balance 'X' in an equation.

Table 3.19: Applying Option Changes State of Equation

Equation	Apply Option	New Equation
x - 2 = 3	Add 2	x - 2 + 2 = 3 + 2
2x + 1 = x	Subtract x	2x - x + 1 = x - x
4 = 1 / x	Multiply x	4 × x = 1 / x × x
3x = 1	Divide 3	3x / 3 = 1 / 3

  - **Applying Simplification in Equation Solving Page:** The simplification can be applied by pressing the simplify button. It will allow the equation to solve like terms and shorten the equation as much as possible as shown in Table 3.20.

Table 3.20: Simplification on Equation Examples

Equation	Simplified Equation
x - 2 + 2 = 3 + 2	x = 5
2x - x + 1 = x - x	x + 1 = 0
4 × x = 1 / x × x	4x = 1
3x / 3 = 1 / 3	x = 1/3

– **Undo/Redo of Action in Equation Solving Page:** The history management module allows a player to switch between the states of equation. The process of undo or redo the action on equation, can be processed by pressing the buttons on Equation Solving page. It will generate previous states as shown in Table 3.21.

Table 3.21: Undo/Redo Effects on Equation

States Number	Current Equation	Redo/Undo	Obtained Equation
1	x + 2 = 3	Undo	x + 2 = 3 (same)
2	x + 2 - 2 = 3 - 2	Undo	x + 2 = 3
3	x + 0 = 5	Redo	x = 5
4	x = 5	Redo	x = 5 (same)

- **Score Display Page:** The selection of right and wrong options will leads to some state of an equation. If this state is a final state, then the system will calculate all the score from selection of right and wrong options and display percentage in Score Display Page. The different marks will be obtained upon selection of right or wrong option as shown in Table 3.22.

Table 3.22: Scores Obtained from Selecting Option

Option Applied	Score Obtained
Right	5
Wrong	-1

4

Conclusion

It is quite challenging and inspiring to design a serious educational game. The game design itself, an interesting and stimulating topic that require full devotion and interest. Thus in this field, the development of serious game needs a great expert and domain knowledge on achieving purpose with entertainment. The developed game has a same concern to achieve goal in the form of fun. The aim was to develop a serious game that helps the student in learning mathematical equations with ease and fun.

In this project, a mathematical game is designed that presented a mathematical equation with related right and wrong options in order to solve equation by keeping the equation balanced on both sides. As it involves resolution of equation by inserting options. Thus the simplification of equation according to every right or wrong option, was a main problem. The insertion of right option shortens the equation while wrong option creates more complex equation. To cater the wrong option and mold it towards solution (although time taking but solve able) is a challenging concern. Besides, keeping motivation level high during a game, balancing fun with learning and removal of boredom are the main concerns while developing the game.

The game is developed on seven modules. Every module has contributed. Level selection module has made a player to choose equation according to its type. This makes a game playable to different students with different concept levels. Equation representation module allows equation to be written in mathematical

form just like written in black board or paper. The game also included options generation for every state of equation so that a player finds all possibilities to solve specific state. Option implementation leads towards solution or sometimes more complex equation depends upon the right option at right time. It provides a sense of similarity as practicing maths on paper. Besides, the option of redo and undo increase this similarity more. A player can goes back or moves to last state of equation. Simplification, the main core of a game is contributed to simplify equation by going through sets of math rules. These rules provide a basic design towards simplification of Algebraic equation. With these basic rules, more complex equations and concepts would be easily developed. For the motivation and learning of player to learn from mistakes, score management has done this job. These modules are the essence of making a base game. As design guidelines, these modules can be used to develop further topics in a game.

The developed game opens an opportunity to bring advancements in the classrooms. Using digital medium as a modern teaching support tool, brings change in learning environment. Students found difficulty in understanding the concept while others are too lazy to practice the problems on paper. Thats why many students hate mathematics but the good thing is almost every student loves game, so putting math as a challenge in a game, could have a potential to make a student starts loving mathematics also. Thus the developed game takes an advantage to initiate this concept:

$$Hate\ Maths\ but\ Love\ Games \Rightarrow Put\ Maths\ in\ Games \Rightarrow Love\ Games\ and\ Maths\ both$$

But beware, sometimes, due to the complex game design the result can be opposite:

$$Hate\ Maths\ but\ Love\ Games \Rightarrow Put\ Maths\ in\ Games \Rightarrow Hate\ Games\ and\ Maths\ both$$

4.1 Future Dimensions

The developed game is a single player playable prototype with an ability to solve mathematical equation that involves one type of variable with degree one. In future, it can improve further both in terms of course content and game application because a project settled the design guidelines and the structure of a framework to enable further developments.

In terms of course contents, further topics would be added i-e second-order equations, trigonometry, linear systems etc. These contents could be inserted by keeping rule-sets as a base for simplification. The rules for additions, subtraction, multiplication and division are always remain the same. Since its a mathematics, the base rules are applicable to any content. Besides, different victory conditions could be imply on a given framework i-e involvement of time factor (the fastest - the better), number of applied options (the shortest - the better) or competing with other players. These are the different but adaptable ways to put interest towards game. Since its a game, the player is always eager to know the reward and performs well to get highest reward. This psychology of a player can be used to put more interest in a game. This eventually puts an interest in mathematics.

In terms of game application, multi mode option can be another advancement to the game in which students in a classroom could compete with one another. For this, a central service can be introduced for controlling the whole gameplay, serving exercises to the students, storing the time, giving high score information to all players. Central service can choose the subjects that will come next and follow the game state, view the high scores, etc. There could be two modes for implementing it. The first mode can be the main application running on a portable device (teachers phone). Specially for class rooms, the teacher would simply run the application and the students access through a mobile device. The second mode can

be to implement an Internet based service with subscription model, that could serve students globally. Besides, winning badges can be introduced. The game can provide different challenges and tasks. The fulfillment of these tasks will makes a player to win different badges. This will creates social motivation and eagerness to get more wins.

Thus with the development of the current game, a set of many and different possibilities has opened. The base design of a game act as a small but an effective step in the field of SEG. Taking the game rules as a design bed, complex mathematical games can be developed in future. More contents, more specifications, new technologies and new interaction designs, will added to make worthy games keeping the current game as a framework. The initiative is to make a digital attractive classroom that is close to student interests and learning. Hence, with improving quality of teaching in the field of education, the game acts as a teaching support tool.

A

Implementation Structure and Examples

The implementation of game modules are done in Javascript language. The game is developed using several Javascripts, HTML pages and CSS styles. Followings are the description of these pages, scripts and styles. Besides, some part of implementation code is also mentioned.

A.1 HTML Pages and CSS Style

The game used HTML 5 version. There are following HTML pages that are involved in game:

- **index.html**: The start-up page for a game to allow players to start a game or learn how to play.
- **how.html**: The page describes how to play a game.
- **syllabus.html**: The page displayed right after the index.html to allow a player to select the level.

- **play.html**: After selection of level, the main page of a game is displayed i-e play.html. This page shows an equation to be solved, list of options to be selected and simplify operation to solve the equation.
- **result.html**: The result is displayed in this page with the information of mistakes that a player done during a game.

Besides, there is also CSS style file i-e **mainStyle.css**. It is used for interfacing and styles in a game.

A.2 Javascripts

The main core of game are Javascript files that are added to implement the modules of a game. There are following main Js files included:

- **main.js**: The main file where the functions of modules are called and work structurally.
- **exercise.js**: The set of equations from different levels are presented in this file.
- **levels.js**: The working of level selection module is operating through this file.
- **expressions.js**: The working of equation representation module, is operating through this file.
- **get_options.js**: The list of right and wrong options are generated by this file. Option generation module is working here.
- **apply_option**: Option implementation module is working through this file. Besides, score is also calculated here when a player applies any option and history management also works when an equation changes its state due to applied option.
- **rules.js**: Simplification of equation is done through this file. If any expression contains subtraction or division property, it converts it into addition and multiplication for further process. It is called in main.js to simplify and do history management, if a simplified equation is different from previous equation.
- **rsum.js**: It includes the set of rules related to addition.
- **rmul.js**: It includes the set of rules related to multiplication.
- **game_test.js/test-examples.js**: These files are used for testing the game modules.
- **jquery.js**: For implementing some jquery functions in a game interface.

A.3 Main Code Examples of Game Modules

The development of a game in terms of implementation is shown. There are some basic and main parts of coding, are mentioned that runs the game. From level selection to score management, the implementation is done using above mentioned javascript files. Followings are the main example codes:

A.3.1 Code Example of Level Selection

```
function getLevel(num){

// Get level from user and store it
storeLevel(num);

}
//-------------------------------------

function storeLevel(lv){

// Save data to the current local store
localStorage.setItem('level', lv);

}
```

Figure A.1: Code Example of Level Selection. From Level.js File

A.3.2 Code Example of Initiating Game

```
function main(){

// Get the selected level
var level = localStorage.getItem('level');

// Get Equation Info from that Level
var eqInfo = setLevelEq(level);

// Equation
e = eqInfo[0];
//Answer Related to Equation
answer = eqInfo[1];

// Insert in array for History Management
arr_his.push(e);
cur = arr_his.length-1;

// To convert sub and div into sum and mul if any
normalize(arr_his[cur]);

// Get related Options
allowOptions(arr_his[cur]);

// Display Equation
display(arr_his[cur]);
}
```

Figure A.2: Code Example of Initiating Game. From Main.js File

A.3.3 Code Example of Options Generation

```
function allowOptions(meq){

// Split Equation
var split = meq.sub_expressions();
var left = split[0];
var right = split[1];

// If needed, reset previous options
resetOpt();

// Set options of left side of equation
setOptions(left);

// Set options of right side of equation
setOptions(right);

// Set the score of every right option
getRightScore(rightoptAct.length);

// Set the score of every wrong option
getWrongOpt(rightopt, rightoptVal, rightoptAct);

// Shuffle both types of options
mergeOpt();

// If still some space left, add dummy options
optionBalancing();

// Display Options
showOpt();

}
```

Figure A.3: Code Example of Options Generation. From Get_option.js File

A.3.4 Code Example of Getting Right Option

```
    function getOptions(e){

    // Check the property of Expression and Generate Options
    // If Positive then Negative sign option generate and vice versa
    // If Expression is of SUM type
    if (e instanceof SUM){
    // Break Expression into Sub expressions
    var subs = e.sub_expressions();
    var a= subs[0];
    var b= subs[1];

    // Negative sign sub expression
    if ((a instanceof UMIN)) {
    // Add value, opposite action and generate option
    rightoptVal.push(a.expr);
    rightoptAct.push(add);
    rightopt.push(assembleOpt(add , a.expr));
    }

    // Positive sign sub expression
    else if ((a instanceof NUM) || (a instanceof VAR)|| (a instanceof MUL)) {
    // Add value, opposite action and generate option
    rightoptVal.push(a);
    rightoptAct.push(subtract);
    rightopt.push(assembleOpt(subtract , a));
    }
    .....
    .....

    // If Expression is of MUL type
    else if (e instanceof MUL){
    // If Division then Multiplication option generate and vice versa
    var subs = e.sub_expressions();
    var a= subs[0];
    var b= subs[1];

    if ((a instanceof UINV)) {
    rightoptVal.push(a);
    rightoptAct.push(mult);
    rightopt.push(assembleOpt(mult , a));
    }

    else if ((a instanceof NUM) || (a instanceof VAR) || (a instanceof UMIN) || (a
instanceof MUL)) {
    rightoptVal.push(a);
    rightoptAct.push(divi);
    rightopt.push(assembleOpt(divi , a));
    }
```

Figure A.4: Code Example of Getting Right Option. Small Part of Code is Shown. From Get_option.js File

A.3.5 Code Example of Getting Wrong Option

```
function getWrongOpt(opt, val, act){

for(var i = 0; i < opt.length; i++){

// Get Wrong Options from Right Options by Inverting Actions of Right Options
// Get Score of every wrong option generated

wrongoptVal.push(val[i]);
wrongoptAct.push(invertOpt(act[i]));
wrongopt.push(assembleOpt(invertOpt(act[i]), val[i]));
wrongScore.push(negativeScore);
}
}
```

Figure A.5: Code Example of Getting Wrong Option. Inverting Action of Right Options. From Get_option.js File

A.3.6 Code Example of Simplification with History Management

```
function simplifyOpt(){

// Split Equation
var x = splitEq(arr_his[cur]);

// Simplify Each Side
var xleft = simplify(x[0]);
var xright = simplify(x[1]);

// After simplifying, Need to check for victory
checkAns(answer, xleft, xright);

// Join Again
var y = joinEq(xleft, xright);

// If Simplification changes state, Then store new state.
// History Managment
if(!equal_expr(y, arr_his[cur])){

arr_his.splice(cur+1, 0, y);
cur = cur + 1;
}

// Display Simplified Equation
display(arr_his[cur]);
allowOptions(arr_his[cur]);

}
```

Figure A.6: Code Example of Simplification with History Management. From Main.js File

A.3.7 Code Example of Checking and Applying SUM and MUL Rules

```
function simplify(e) {
var s = rule_sum(e);
s = rule_mul(s);
return s;
}

//-----------------------------------------------
// Check and Apply Rules for Addition
function rule_sum(e) {

if (e instanceof SUM) {
var subs = e.sub_expressions();
var a= subs[0];
var b= subs[1];

if (check_rs1(a,b)) {
return apply_rs1(a, b);
}
else if (check_rs2(a,b)) {
return apply_rs2(a, b);
}
else if (check_rs3(a,b)){
return apply_rs3(a, b);
}
else if (check_rs4(a,b)) {
return apply_rs4();
}
.... }

//-----------------------------------------------
// Check and Apply Rules for Multiplication
function rule_mul(e) {

if (e instanceof MUL) {

var subs = e.sub_expressions();
var a= subs[0];
var b= subs[1];

if (check_rm1(a,b)) {
return apply_rm1(a,b);
}
else if (check_rm2(a,b)) {
return apply_rm2(e);
}
else if (check_rm3(a,b)) {
return apply_rm3(a,b);
}
..... }
```

Figure A.7: Code Example of Checking and Applying SUM and MUL Rules. Contains Some Part of Code. From Rules.js File

A.3.8 Code Example of Converting SUB and DIV into SUM and MUL

```
    // Rules for Coverting SUB and DIV into SUM and MUL to Pass through set of
rules of SUM and MUL.

    function rule_sub(e) {
    if (e instanceof SUB) {

    var subs = e.sub_expressions();
    var a= subs[0];
    var b= subs[1];
    return rule_sum(sum( a, umin(b)));
    } else {
    return e; } }

    function rule_div(e) {
    if (e instanceof DIV) {

    var subs = e.sub_expressions();
    var a= subs[0];
    var b= subs[1];
    return rule_mul(mul( a, uinv(b)));
    } else {
    return e; } }
```

Figure A.8: Code Example of Converting SUB and DIV into SUM and MUL. From Rules.js File

A.3.9 Code Example of SUM Rules

```
    function check_rs1(a,b){
    // Both are positive numbers
    return (checkNum(a)==pnum) && (checkNum(b)==pnum);
    }

    function apply_rs1(num1, num2){
    // return result of adding two numbers
    return num(num1.expr + num2.expr);
    }

    //-----------------------------------------

    function check_rs2(a,b){
    // Both are positive variables without coefficient
    return (checkVar(a)==pvar) && (checkVar(b)==pvar);
    }

    function apply_rs2(var1, var2){
    // return result of adding x + x
    return rule_mul(mul(num(2), var_(var1.expr)));
    }
```

Figure A.9: Code Example of SUM Rules. Few are Mentioned. From Rsum.js File

A.3.10 Code Example of MUL Rules

```
function check_rm1(a,b){

// Both are positive numbers
return (checkNum(a)==pnum) && (checkNum(b)==pnum);
}
function apply_rm1(num1, num2){

return num(num1.expr * num2.expr);
}

...

function check_rm5(a,b){

// both are negative num
return (checkNum(a)==nnum) && (checkNum(b)==nnum);
}

function apply_rm5(a,b){

return num( a.expr.expr * b.expr.expr);

}

...

function check_rm7(a,b){

// First is Positive Var and other is Negative Num
return (checkVar(a)==pvar) && (checkNum(b)==nnum);
}

function apply_rm7(a,b){

// If x and 0
if(b.expr==0){
return num(0);
}

// If x and -1
else if(b.expr==1){
return umin(var_(a.expr));
}

// If x and -num
else{
return umin(mul(b.expr,a));
}

}
```

Figure A.10: Code Example of MUL Rules. Few are Mentioned. From Rmul.js File

A.3.11 Code Example of Option Implementation with Score

```
    function apply(bt){

    // To Get Ideal Score for Percentage
Calculation
    perfect_score = perfect_score +
positiveScore;

    // Apply Option (Value, Action, Index of
Option)
    applyOption(optionsVal, optionsAct, bt);

    // Add Score of Applying specific option
    addScore(bt);

    // Display Equation After Applied Options
    display(arr_his[cur]);
```

Figure A.11: Code Example of Option Implementation with Score Management. From Apply_options.js File

Apply Option Function

```
//TO APPLY OPTIONS
function applyOption(optVal, optAct, index){

if(index<=optAct.length){

// Matching Action with Option Actions

if(optAct[index]==add){
arr_his.splice(cur+1, 0, arr_his[cur].add_bothsides(optVal[index]));
cur = cur + 1;
}
else if(optAct[index]==subtract){
arr_his.splice(cur+1, 0, arr_his[cur].sub_bothsides(optVal[index]));
cur = cur + 1;
}
else if(optAct[index]==mult){
arr_his.splice(cur+1, 0, arr_his[cur].mul_bothsides(optVal[index]));
cur = cur + 1;
}
else if(optAct[index]==divi){
arr_his.splice(cur+1, 0, arr_his[cur].div_bothsides(optVal[index]));
cur = cur + 1;
} } }
```

Figure A.12: Code Example of Apply Option by Matching Action of Option. History Management Also Involved From Apply_options.js File

A.3.12 Code Example of Equation Representation

```
function EQ() {
Expression.call(this);
this.op = "EQ";
this.leftside = "";
this.rightside = "";
this.subexpr = [this.leftside, this.rightside];
}
EQ.prototype = new Expression();
EQ.prototype.constructor = EQ;
EQ.prototype.init = function(leftside,rightside)  {
this.leftside = leftside;
this.rightside = rightside;
this.subexpr= [this.leftside, this.rightside];
return this;
}
// Representing Equation
EQ.prototype.to_LaTeX = function() {
return this.leftside.to_LaTeX() + " = " + this.rightside.to_LaTeX();
}

// Adding Option on both sides
EQ.prototype.add_bothsides = function(x) {
s = eq(this.leftside, this.rightside);
this.leftside = sum(this.leftside,x);
this.rightside = sum(this.rightside,x);
return eq(this.leftside, this.rightside);
}

// Subtracting Option on both sides
EQ.prototype.sub_bothsides = function(x) {
s = eq(this.leftside, this.rightside);
this.leftside = sum(this.leftside,umin(x));
this.rightside = sum(this.rightside,umin(x));
return eq(this.leftside, this.rightside);
}

// Multiplying Option on both sides
EQ.prototype.mul_bothsides = function(x) {
s = eq(this.leftside, this.rightside);
this.leftside = mul(this.leftside,x);
this.rightside = mul(this.rightside,x);
return eq(this.leftside, this.rightside);
}

// Divide Option on both sides
EQ.prototype.div_bothsides = function(x) {
s = eq(this.leftside, this.rightside);
this.leftside = mul(this.leftside,uinv(x));
this.rightside = mul(this.rightside,uinv(x));
return eq(this.leftside, this.rightside);
}
function eq(a,b) {
return new EQ().init(a,b);
}
```

Figure A.13: Code Example of Applying Option on Both Sides of Equation. Representation of Equation. From Expressions.js File

A.3.13 Code Example of Expression Representation

```
//These are generic math expressions
//General View

var s = [];
function Expression() {

this.op="Expression";
this.expr = "";
this.subexpr = [];

}

Expression.prototype.constructor = Expression;
Expression.prototype.init = function(x)  {

this.expr = x;
return this;

}

// Representing Expression in Mathematical Form
Expression.prototype.to_LaTeX = function() {

return this.expr;

}

// Obtaining Sub Expressions
Expression.prototype.sub_expressions = function() {

return this.subexpr;

}

// Converting To String
Expression.prototype.toString = function() {

var str_subexpr = "";
for (var i in this.subexpr) {

str_subexpr = str_subexpr.concat(this.subexpr[i].toString()).concat(",");

}

return this.op.concat("(").concat(str_subexpr).concat(")");

}
```

Figure A.14: Code Example of Representing an Expression. Generalized View. From Expressions.js File

A.3.14 Representing MUL Expression. Specific View Example

```
        function MUL() {
        Expression.call(this);
        this.op = "MUL";
        this.a = "";
        this.b = "";
        this.subexpr= [this.a, this.b];
        }
        MUL.prototype = new Expression();
        MUL.prototype.constructor = MUL;
        MUL.prototype.init = function(a,b) {
        this.a = a;
        this.b = b;
        this.subexpr= [this.a, this.b];
        return this;
        }
        MUL.prototype.to_LaTeX = function() {
        if ( ((this.a instanceof NUM) && (this.b instanceof VAR)) || (this.b
    instanceof UINV) ) {
        return this.a.to_LaTeX() + this.b.to_LaTeX();
        }

        else if (this.a instanceof UINV){

        return this.b.to_LaTeX() + this.a.to_LaTeX();

        }
        else {
        return this.a.to_LaTeX() + "\\times " + this.b.to_LaTeX();
        }
        }

        function mul(a,b) {
        return new MUL().init(a,b);
        }
```

Figure A.15: Code Example of Representing MUL Expression. Specific View Example. Some Part of Code. From Expressions.js File

B

Demonstration of Running Game

The game has run on a laptop to collect screen shots of a running game in order to give demonstration. It contains a flow of game that how a game will proceed. A game is consist of several pages that come on a screen step by step. The player has to select type of equation to be solved. Then the selected equation will be presented to the player for solving. Equation will be presented with options that a player can choose to solve it. The options will be given again and again according to the states of equation. When a player reaches to solution, the game will ends and displays the results. Followings are the steps or sequences presented here:

B.1 Starting To Play:

Starting the game shows several options. Choose the option to play a game as shown in Figure B.1. This will leads to level selection.

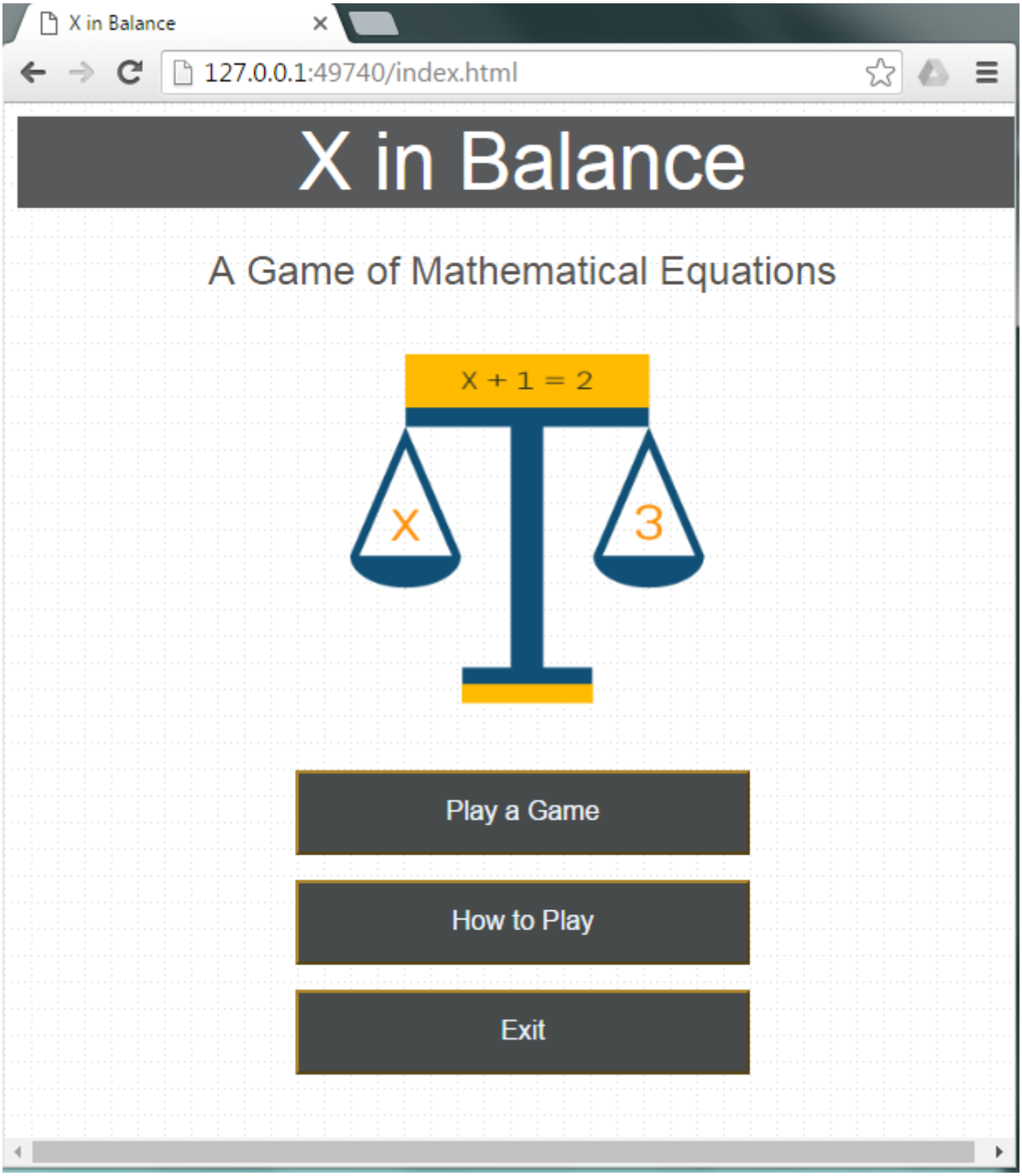

Figure B.1: Starts to Play. The figure shows a screen shot from running game.

B.2 Selecting Level:

Choose the level and proceeds to solving of equation as shown in Figure B.2. Easy level is a basic level to start knowing the concept. Medium level is to practice equations while hard level is for mastering the complex equations.

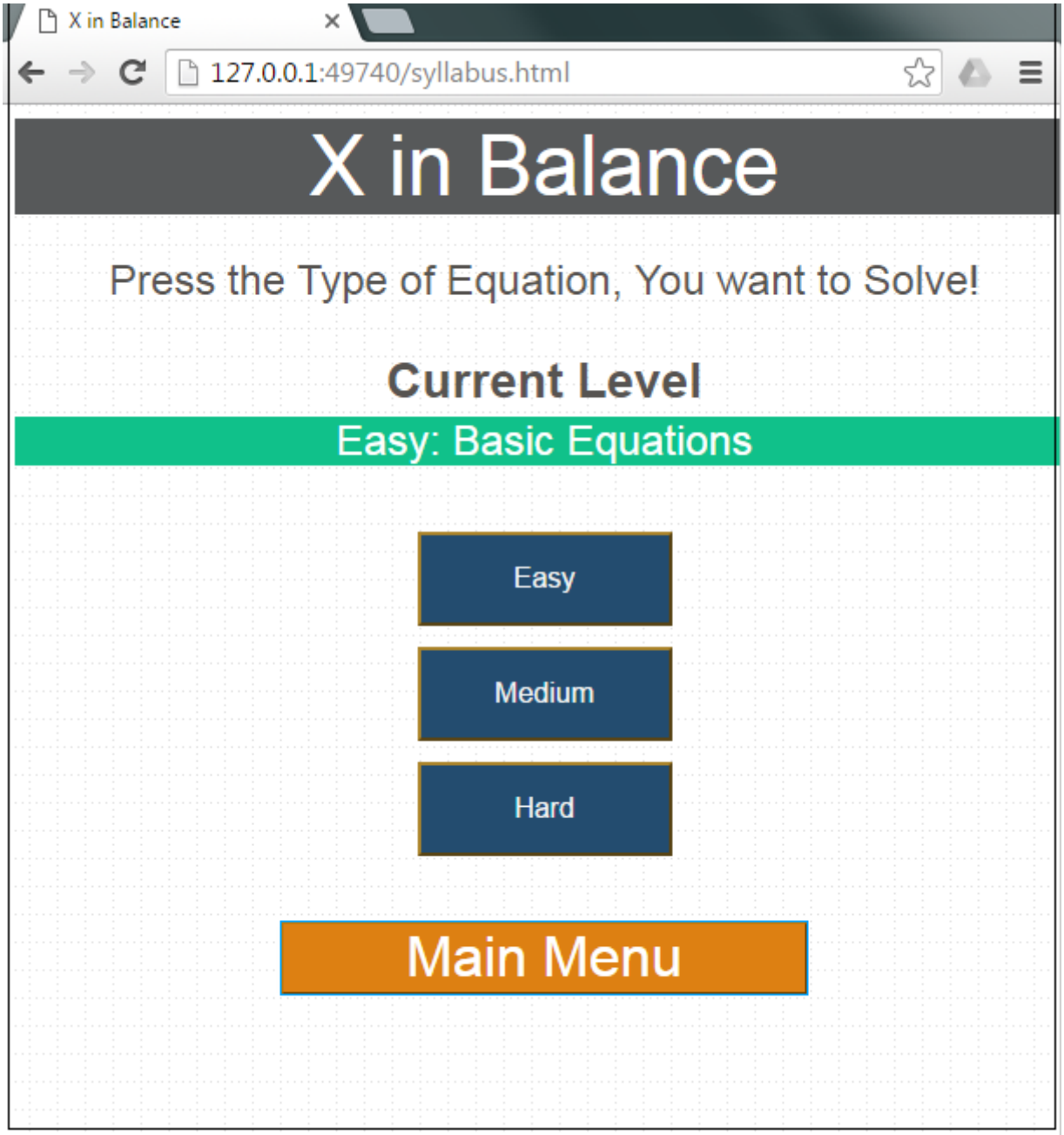

Figure B.2: Choose the Level. The figure shows a screen shot from running game.

B.3 Solving Equation:

After choosing level, an equation is chosen randomly from that level for solving. With the equation, a set of right and wrong options are displayed as shown in Figure B.3. Both the options are shuffled. There are total nine options displayed on a screen. The player is able to select any of the options. Besides, there are three buttons that will do the working of undo, redo and simplify, if necessary.

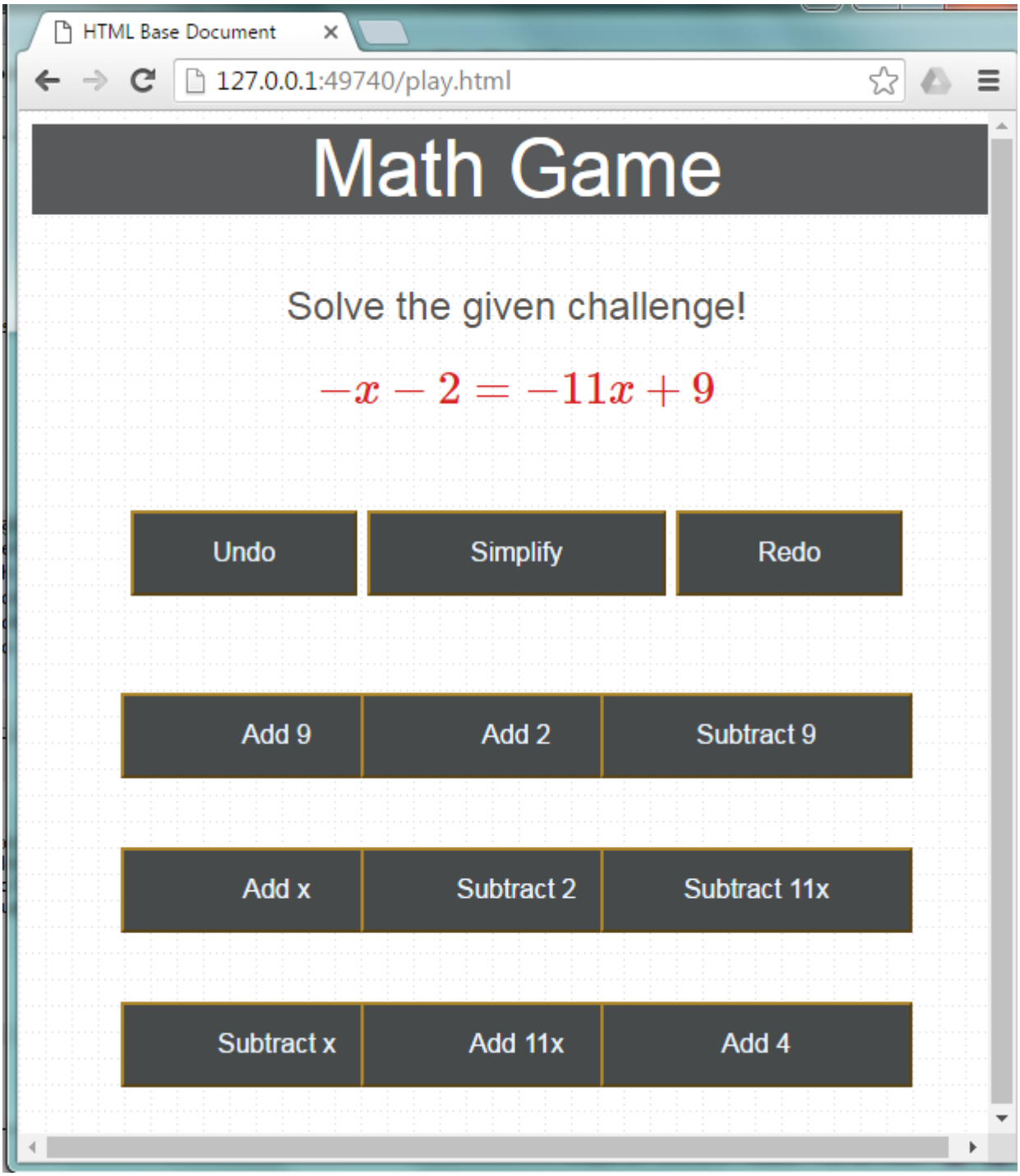

Figure B.3: Solving Equation. The figure shows a screen shot from running game.

B.4 Apply Option:

The selection of option button. changes the state of equation as shown in Figures B.4 and B.5. Adding options to current equation will leads to balance 'X' in an equation.

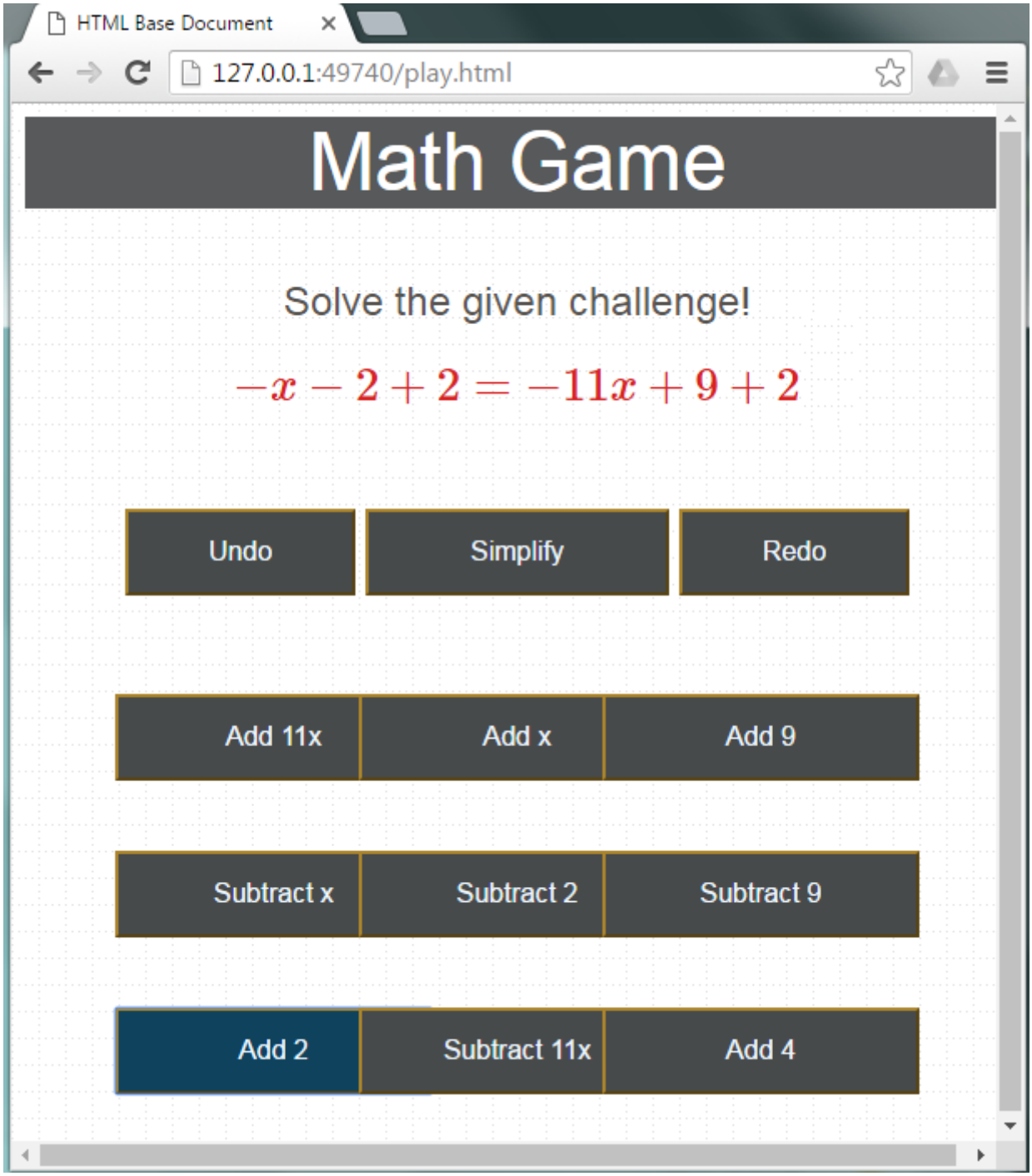

Figure B.4: Applying Right Option. The figure shows a screen shot from running game.

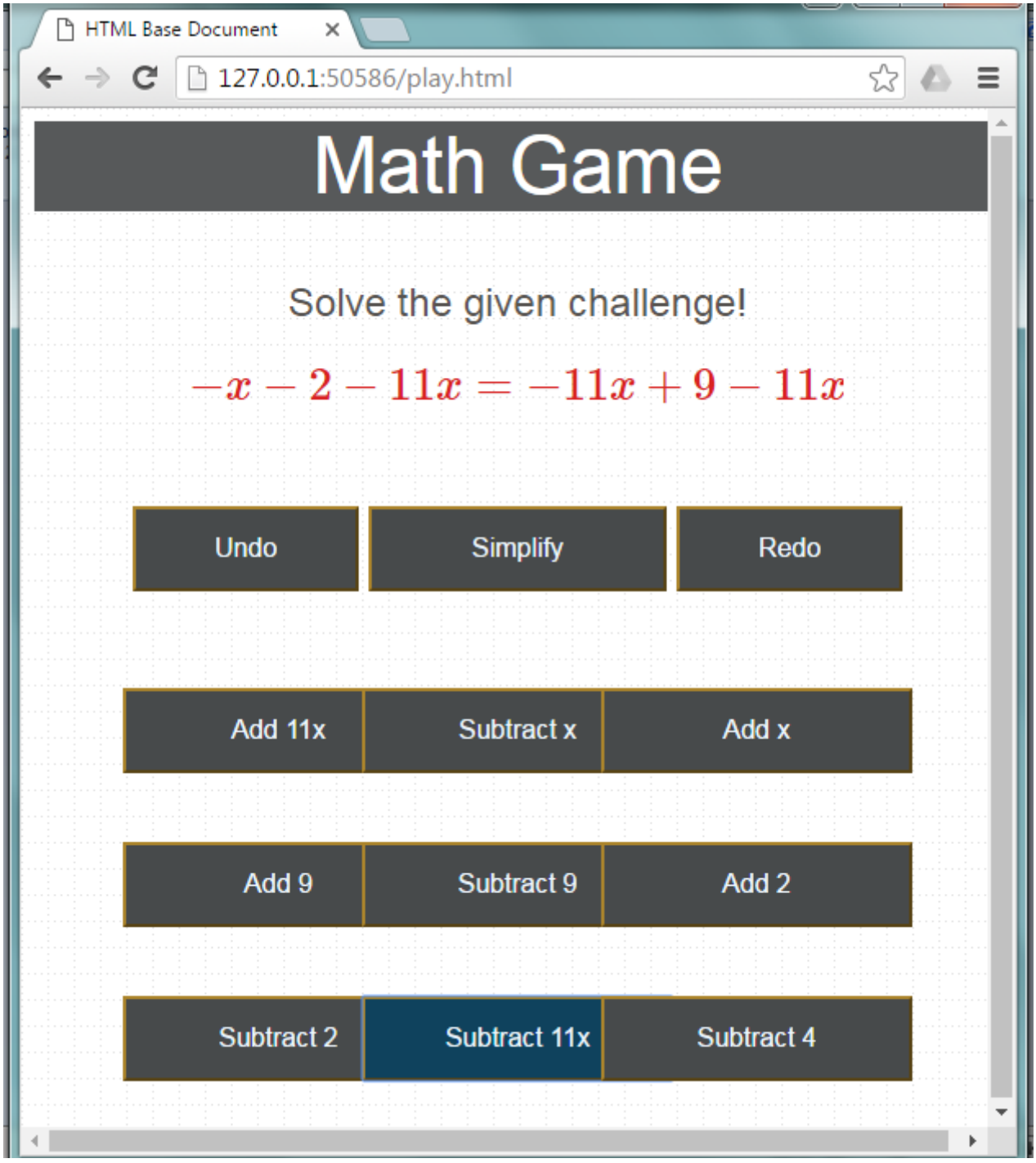

Figure B.5: Applying Wrong Option. The figure shows a screen shot from running game.

B.5 Apply Simplification:

The simplification can be applied by pressing the simplify button as shown in Figure B.6. It will allow the equation to solve like terms and shorten the equation as much as possible. While equation with applied

wrong option could not simplified to short one, in fact, it increases the work (enlarges equation) as shown in Figure B.7.

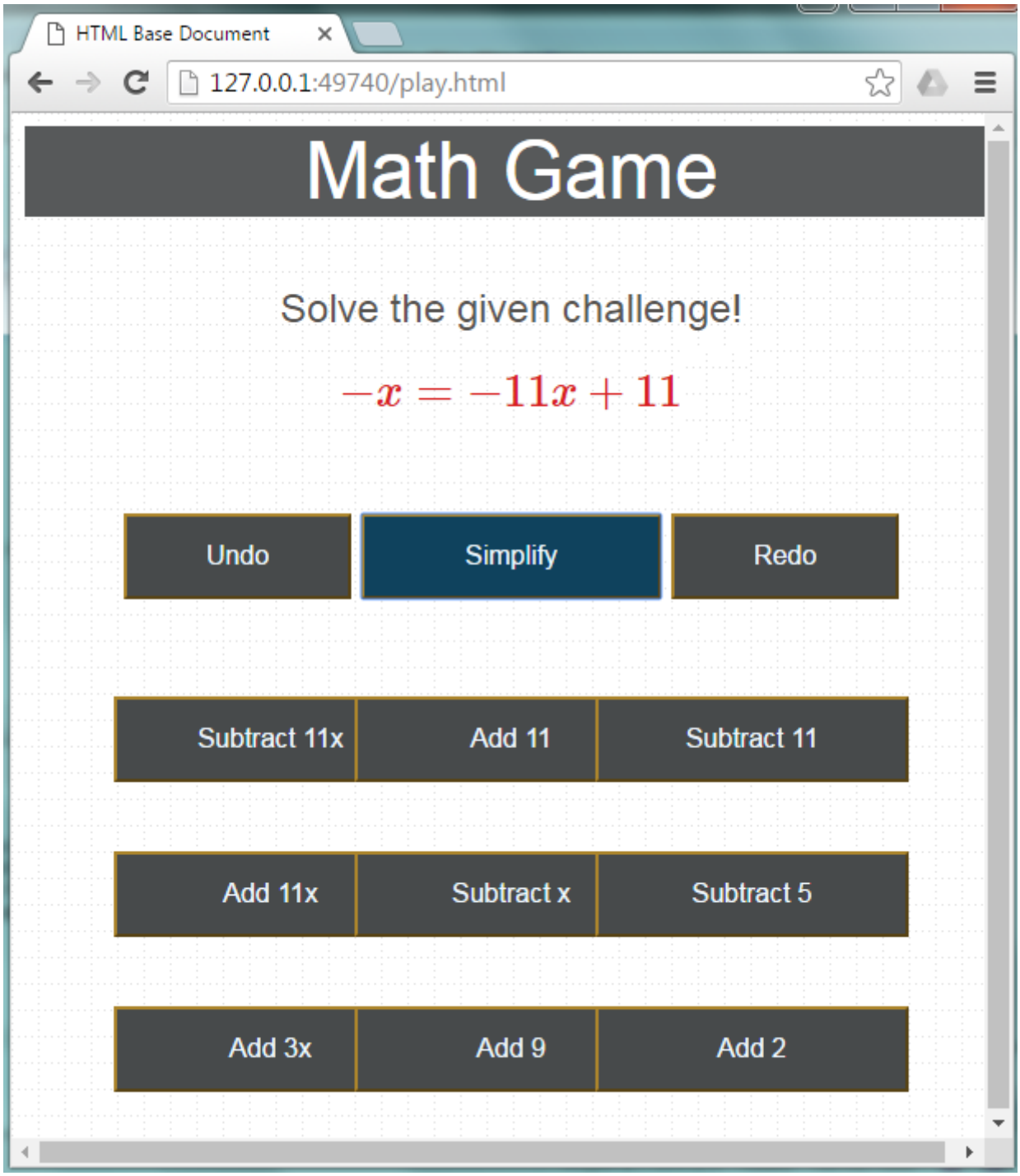

Figure B.6: Applying Simplification on Equation with Right Option. The figure shows a screen shot from running game.

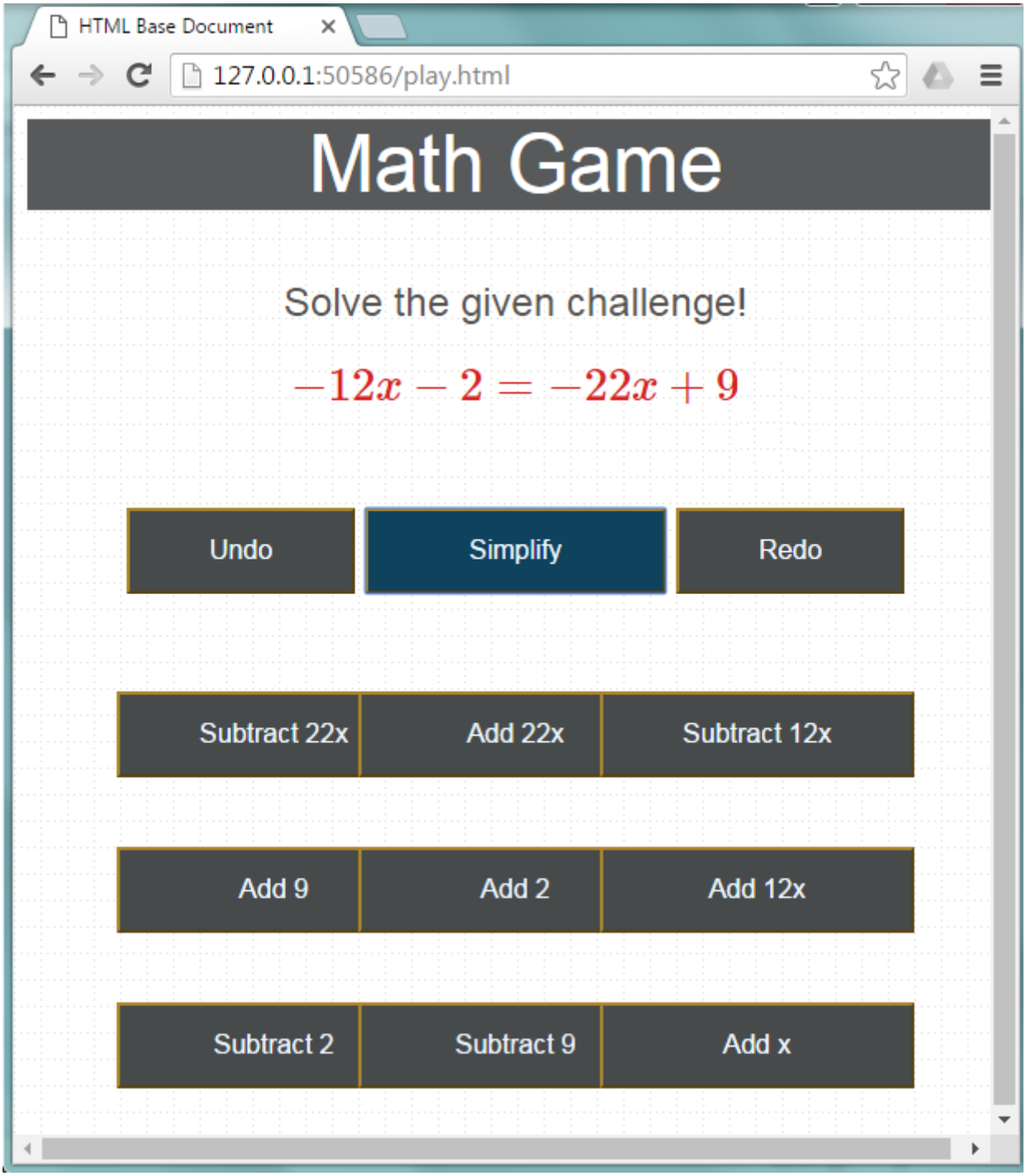

Figure B.7: Applying Simplification on Equation with Wrong Option. The figure shows a screen shot from running game.

B.6 Undo/Redo of Action:

The history management module allows a player to switch between the states of equation. The process of undo or redo the action on equation, can be processed by pressing the buttons as shown in Figures B.8

and B.9.

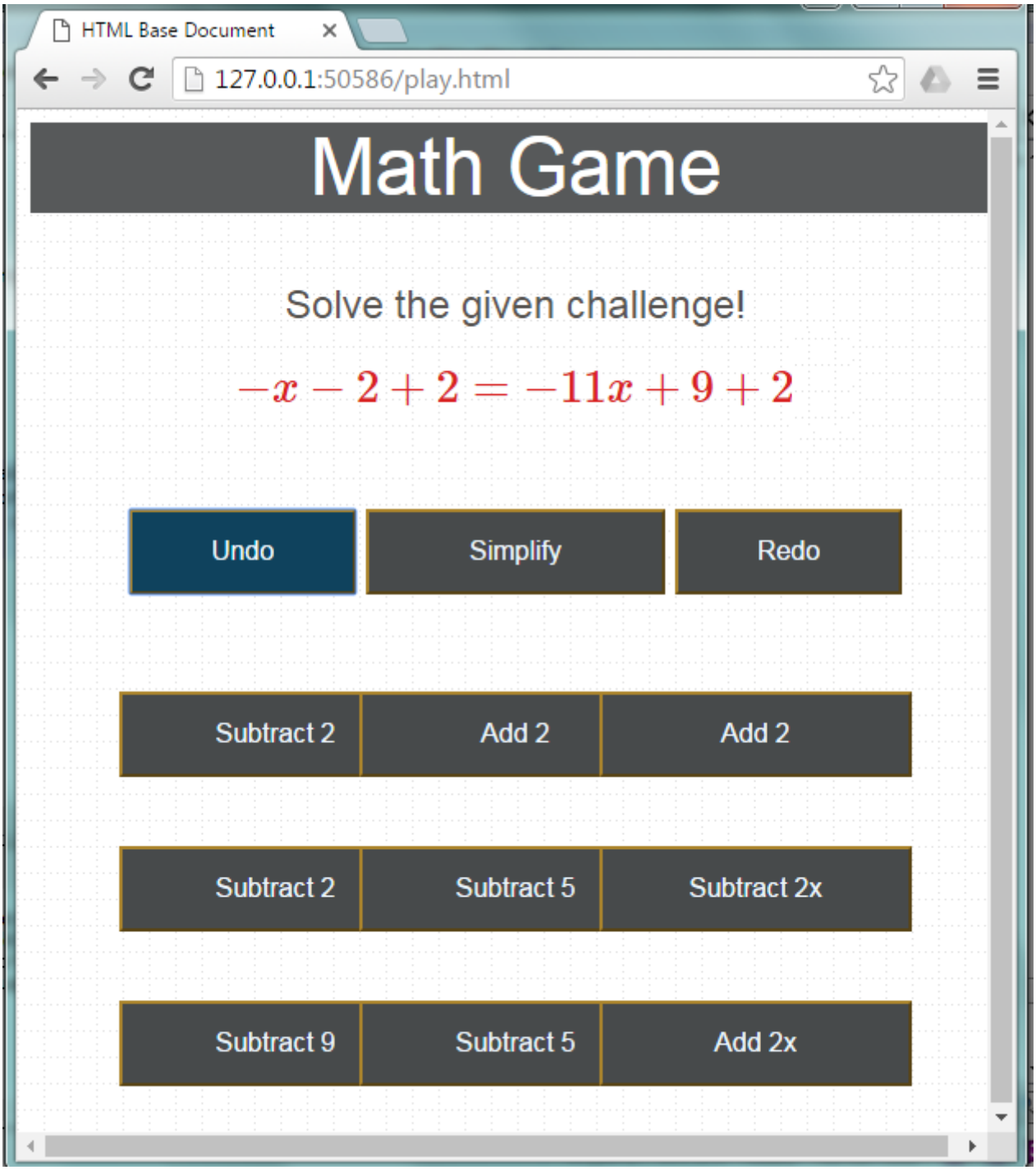

Figure B.8: Applying Undo to Equation (To which right option was applied). The figure shows a screen shot from running game.

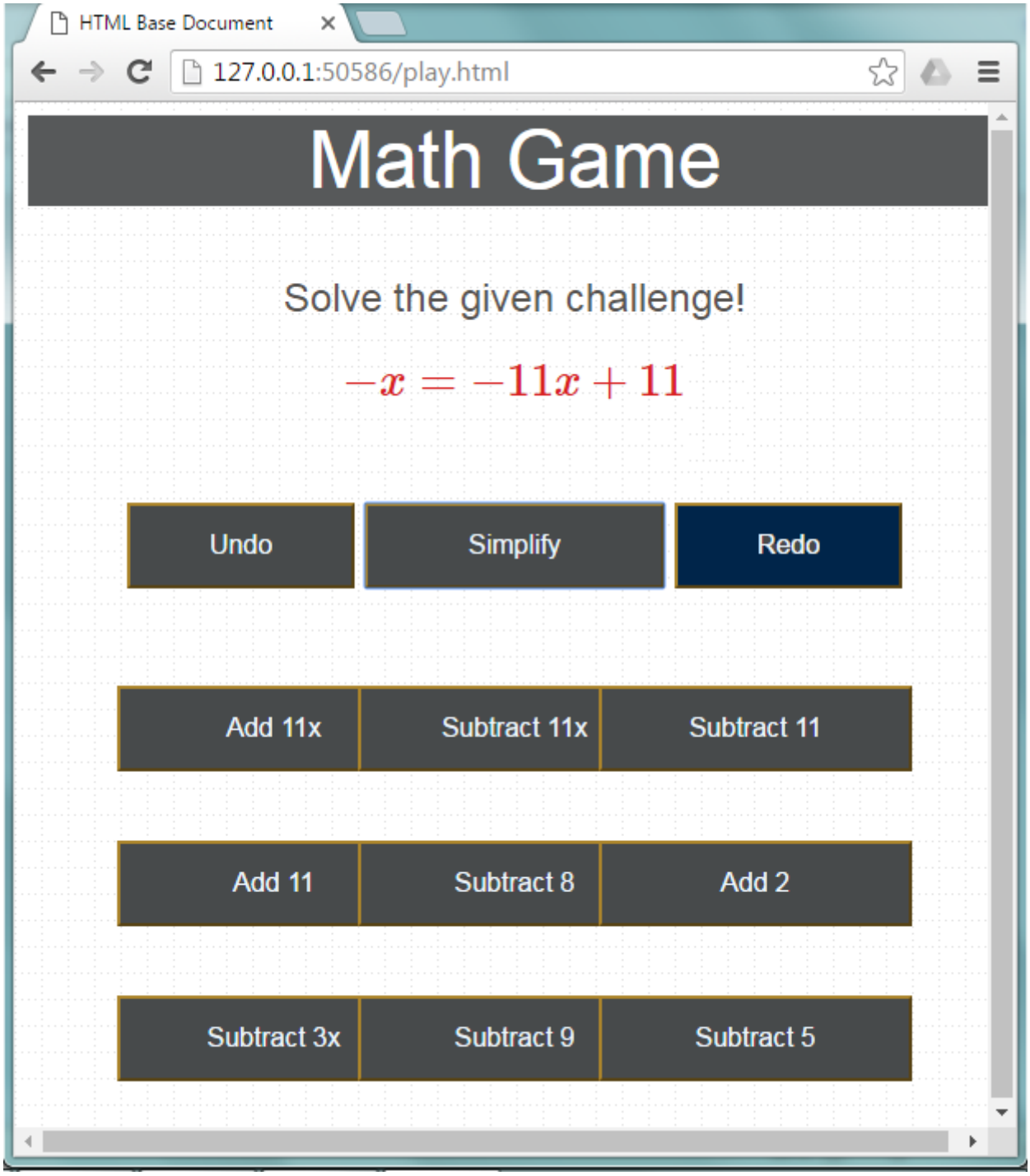

Figure B.9: Applying Redo to Equation (To which right option was applied). The figure shows a screen shot from running game.

B.7 Play Until Reached

With every state of equation, options will change. A player should selects one option to apply on equation. The implementation of applied option will changes to new state. The player keeps on selecting option until

a final state will reached i-e solution of equation as shown in Figure B.10.

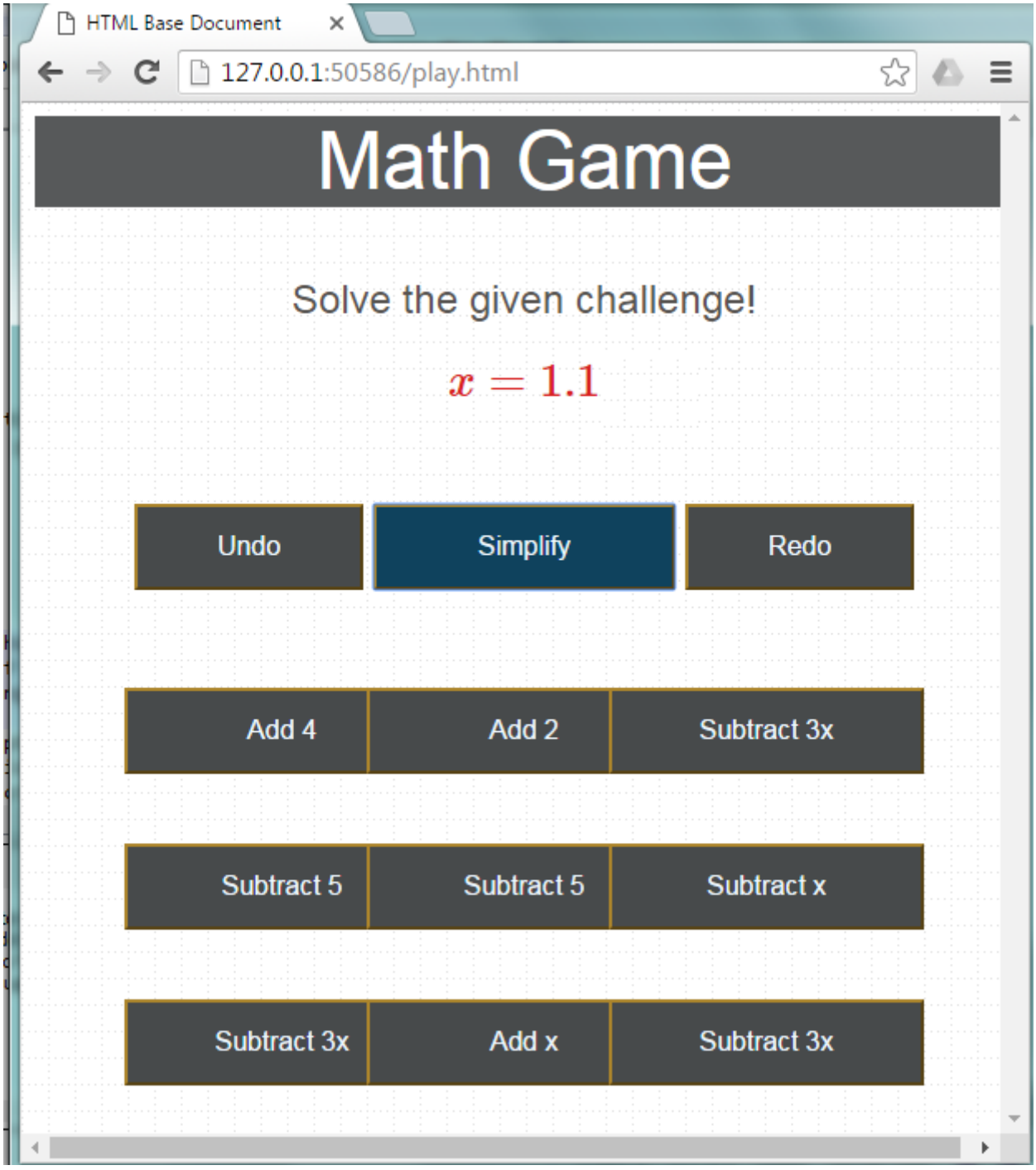

Figure B.10: Final State of Equation (Solution). The figure shows a screen shot from running game.

B.8 Score of Player:

The selection of right and wrong options will leads to some state of an equation. If this state is a final state, then the system will calculate all the score from selection of right and wrong options and display percentage as shown in Figure B.11.

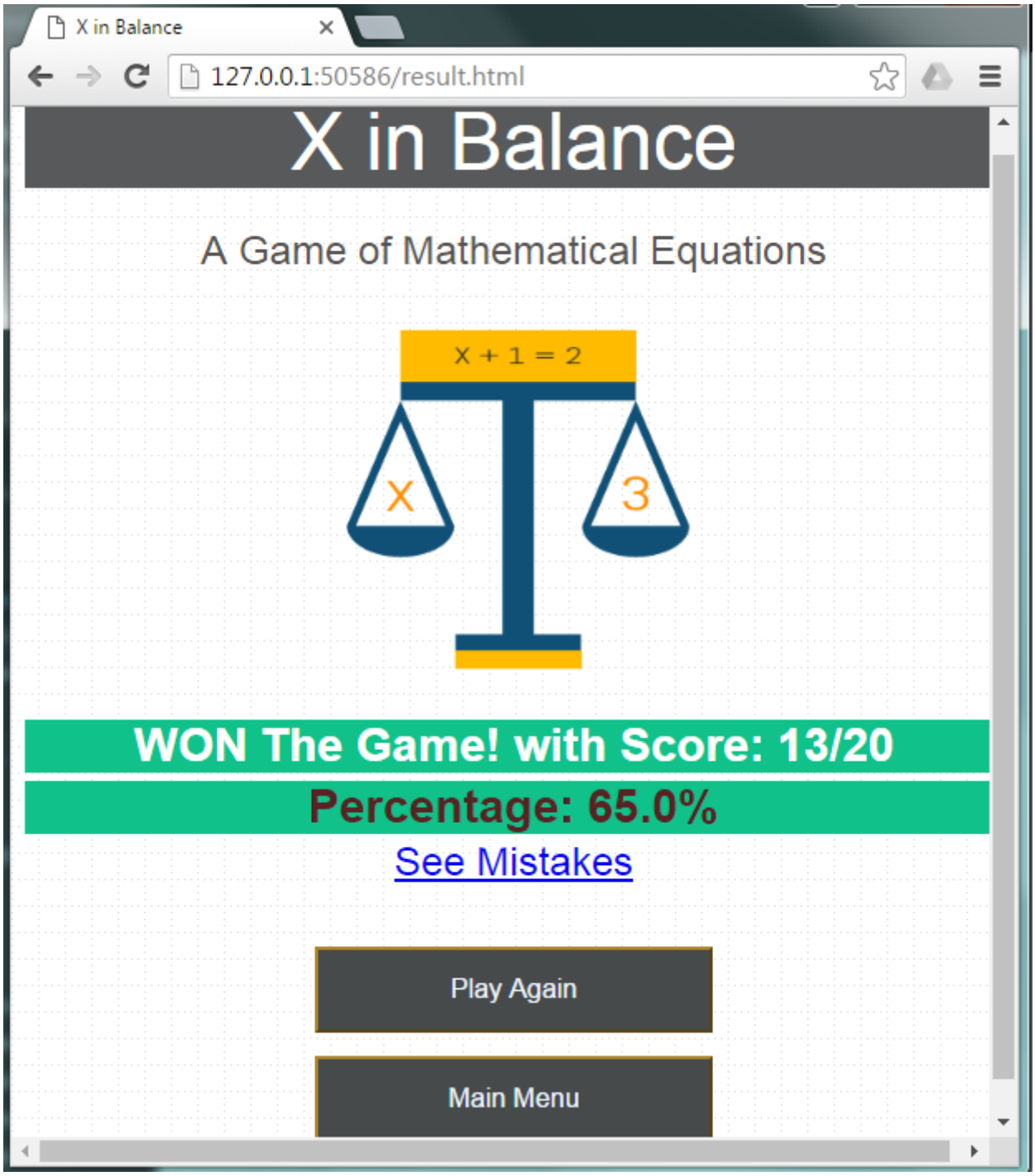

Figure B.11: Getting Final Score. The figure shows a screen shot from running game.

Bibliography

[air] Persuasive games -airport insecurity. `http://www.persuasivegames.com/games/game.aspx?game=airportinsecurity`. [Online; accessed 20-May-2016].

[APD14] Floor Aarnoutse, Linda Peursum, and Fabiano Dalpiaz. The evolution of advergames development: A study in the netherlands. In *Games Media Entertainment (GEM), 2014 IEEE*, pages 1–8. IEEE, 2014.

[BAAH13] Fran C Blumberg, Debby E Almonte, Jared S Anthony, and Naoko Hashimoto. Serious games: What are they? what do they do? why should we play them. *The Oxford Handbook of Media Psychology, Oxford, Oxford Library of Psychology*, pages 334–351, 2013.

[BB10] Johannes S Breuer and Gary Bente. Why so serious? on the relation of serious games and learning. *Eludamos. Journal for Computer Game Culture*, 4(1):7–24, 2010.

[BH13] P Backlund and Maurice Hendrix. Educational Games–Are They Worth The Effort? *5th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES)*, pages 1–8, 2013.

[BS09] Brenda Brathwaite and Ian Schreiber. *Challenges for game designers*. Nelson Education, 2009.

[CK10] Lorenzo Cantoni and Nadzeya Kalbaska. The waiter game: structure and development of an hospitality training game. In *Games and Virtual Worlds for Serious Applications (VS-GAMES), 2010 Second International Conference on*, pages 83–86. IEEE, 2010.

[dar] Darfur is dying - play mtvu's darfur refugee game for change. `http://www.darfurisdying.com/aboutgame.html`. [Online; accessed 29-May-2016].

[FC14] Frederico Ferreira and Sofia Cavaco. Mathematics for all: a game-based learning environment for visually impaired students. In *Frontiers in Education Conference (FIE), 2014 IEEE*, pages 1–8. IEEE, 2014.

[GCJ12] Michail Giannakos, Konstantinos Chorianopoulos, and Letizia Jaccheri. Math is not only for science geeks: Design and assessment of a storytelling serious video game. In *Advanced Learning Technologies (ICALT), 2012 IEEE 12th International Conference on*, pages 418–419. IEEE, 2012.

[GSB14] Jing Guo, Nicolas Singer, and Rémi Bastide. Design of a serious game in training non-clinical skills for professionals in health care area. In *Serious Games and Applications for Health (SeGAH), 2014 IEEE 3rd International Conference on*, pages 1–6. IEEE, 2014.

[HCSB11] Thomas Hainey, Thomas M Connolly, Mark Stansfield, and Elizabeth A Boyle. Evaluation of a game to teach requirements collection and analysis in software engineering at tertiary education level. *Computers & Education*, 56(1):21–35, 2011.

[KC10] Sunha Kim and Mido Chang. Computer games for the math achievement of diverse students. *Educational Technology & Society*, 13(3):224–232, 2010.

[Leo12] Lisa Leopold. Prewriting tasks for auditory, visual, and kinesthetic learners. *TESL Canada Journal*, 29(2):96–102, 2012.

[LPB+15] Jose Eduardo Nunes Lino, Marco Antonio Paludo, Fabio Vinicius Binder, Sheila Reinehr, and Andreia Malucelli. Project management game 2d (pmg-2d): A serious game to assist software project managers training. In *Frontiers in Education Conference (FIE), 2015. 32614 2015. IEEE*, pages 1–8. IEEE, 2015.

[Mac13] RF Mackay. Playing to learn: Panelists at stanford discussion say using games as an educational tool provides opportunities for deeper learning. *Stanford News. Retrieved August*, 21:2013, 2013.

[mcv] Mcdonald's video game. `http://www.mcvideogame.com/index-eng.html`. [Online; accessed 29-May-2016].

[MTJV09] Mathieu Muratet, Patrice Torguet, Jean-Pierre Jessel, and Fabienne Viallet. Towards a serious game to help students learn computer programming. *International Journal of Computer Games Technology*, 2009:3, 2009.

[PMP10] Fotini Paraskeva, Sofia Mysirlaki, and Aikaterini Papagianni. Multiplayer online games as educational tools: Facing new challenges in learning. *Computers & Education*, 54(2):498–505, 2010.

[rem] Re-mission 2: Fight cancer and win!. games for cancer support based on scientific research. `http://www.re-mission2.org/`. [Online; accessed 30-May-2016].

[scr] Scratch - imagine, program, share. `https://scratch.mit.edu/`. [Online; accessed 28-May-2016].

[SMP15] Pedro Sequeira, Francisco S Melo, and Ana Paiva. "let's save resources!": A dynamic, collaborative ai for a multiplayer environmental awareness game. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 399–406. IEEE, 2015.

[TBF15] Yassine Tazouti, Siham Boulaknadel, and Youssef Fakhri. Enhancing young children's mathematics skills using serious game approach. In *2015 5th International Conference on Information & Communication Technology and Accessibility (ICTA)*, pages 1–3. IEEE, 2015.

[TFDW11] Sigmund Tobias, JD Fletcher, David Yun Dai, and Alexander P Wind. Review of research on computer games. *Computer games and instruction*, 127:222, 2011.

[VVCB+06] Jennifer J Vogel, David S Vogel, Jan Cannon-Bowers, Clint A Bowers, Kathryn Muse, and Michelle Wright. Computer gaming and interactive simulations for learning: A meta-analysis. *Journal of Educational Computing Research*, 34(3):229–243, 2006.

[Wik] Wikipedia. Big bumpin'. `https://en.wikipedia.org/wiki/Big_Bumpin%27`. [Online; accessed 24-May-2016].

[WJK08] Monica Wijers, Vincent Jonker, and Kristel Kerstens. Mobilemath: the phone, the game and the math. In *Proceedings of the European Conference on Game Based Learning, Barcelona*, pages 507–516, 2008.

[Yan12] Ya-Ting Carolyn Yang. Building virtual cities, inspiring intelligent citizens: Digital games for developing students' problem solving and learning motivation. *Computers & Education*, 59(2):365–377, 2012.

[Zyd05] Michael Zyda. From visual simulation to virtual reality to games. *Computer*, 38(9):25–32, 2005.

Index

UNIVERSIDADE DE ÉVORA
INSTITUTO DE INVESTIGAÇÃO
E FORMAÇÃO AVANÇADA

Contactos:
Universidade de Évora
Instituto de Investigação e Formação Avançada — IIFA
Palácio do Vimioso | Largo Marquês de Marialva, Apart. 94
7002 - 554 Évora | Portugal
Tel: (+351) 266 706 581
Fax: (+351) 266 744 677
email: iifa@uevora.pt