December 23rd, 2012

Department of Information Systems

**BGU**

**PRODIGY**

**A Computer Game Enhanced E-Learning System**

Academic Instructor

**Dr. Meirav Taieb-Maimon**

By

**Guy Manzurola** (037769726)

ABSTRACT

In almost all areas of learning, knowledge is acquired through extensive and repetitive practice of the learned material. This is especially true when learning a second language.

A popular and appealing course of providing users with learning material is via digital devices, i.e. “e-learning”. This enables learners to study at a time and pace of their choosing.

However, most if not all of the existing e-learning solutions focus on the needs of educators to impart “selected knowledge” to their students, rather than fulfilling the needs of the learners (Pivec, 2007; McGinnis, Bustard, Black, & Charles, 2008; Jong, Shang, Lee, & Law, 2006).

This leaves the learners to find the necessary motivation to study the content themselves.

The purpose of this project is to develop an e-learning quiz system using design principles from computer games, in order to keep users engaged and motivated through repetitive practice. This work will focus on English language structure learning content.

LITERATURE REVIEW

It is known that a brain enjoying itself functions more efficiently. In other words, if we enjoy learning, we learn better.

Based on the above, it appears that there is a long term need for a tool that would make the learning process engaging and motivating for the learner.

1. E-Learning
   1. What is E-Learning?

In essence, **E-Learning is** **the transfer of instructional material through digital devices** such as mobile phones, tablet computers, laptop and desktop computers, and more .

E-Learning enables learners to study at times and places of their choosing. It may also provide people with the ability to study at their own pace (asynchronous study), and in the order they prefer. It is especially appealing now days, due to the high availability of the instruments required to participate in such activities (Clark & Mayer, 2011; McGinnis, Bustard, Black, & Charles, 2008).

Tools of this type can be used either to complement and support traditional teaching, or as a complete and independent solution that can replace physical teacher-student interaction all together. The latter is much less common, and where found, is mainly used as a training tool for specific skills (e.g. tutorial on a piece of software). The former however is used by numerous institutions, either to consolidate learning material, or to present students with practical exercises relevant to the learnt material.

* 1. Problems with Current E-Learning Tools

Though e-learning has great potential to open up a number of different opportunities for learners around the globe (Smith, 2008), most E-Learning systems do not yet have the impact that many believe is possible (McGinnis, Bustard, Black, & Charles, 2008). Without the proper motivation for students to engage in a learning experience, e-learning initiatives following this trend are likely to fail (Smith, 2008). The cause of the problem can be attributed to the following two factors.

* + 1. Meeting Learners Needs

Current e-learning systems tend to focus on the management and delivery of content (McGinnis, Bustard, Black, & Charles, 2008; Pivec, 2007), which is one of the primary goals of an educator (to instruct) (Prensky, 2002). Thus a situation is created, where the importance of the student within the process is often overlooked; the focus is mainly on the needs of educators, instead of the needs of the learners who are actually using the system to learn.

A learner’s needs may be presented by following three demands (McGinnis, Bustard, Black, & Charles, 2008; Pivec, 2007):

1. *Empowerment*. Learners expect to be in control of the learning experience. E-learning systems should thus promote self-directed learning.
2. *Social Identity*. The freedom a learner has in a self-directed learning environment should not come at the expense of a sense of belonging to a social group. Research indicates that such a sense improves motivation and effective learning.
3. *Authentic Learning Experience*. Learners are more engaged and motivated when they are participating in activities to which they may relate.
   * 1. Lack of Compelling Content

**Engaging learners long enough to see them through to the end of a course has become one of the most significant problems faced by e-learning developers**. This can be attributed to a lack in one or more of three main issues as suggested by McGinnis et al (McGinnis, Bustard, Black, & Charles, 2008):

1. *Interaction*. Immediate feedback, clear short term goals and a sense of ‘flow’ through the content is needed for an effective E-Learning experience.
2. *Challenge*. Unchallenging material has been indicated by learners as not motivating, which results in many learners who are reluctant to repeat the experience.
3. *Context*. Students have stressed the importance of being able to appreciate the significance of their current progress in relation to the overall goal of the learning material.

**Without compelling content, students are left to find the necessary motivation themselves.**

1. Enhancing Motivation Through Games

**It is known that a brain enjoying itself functions more efficiently. In other words, if we enjoy learning, we learn better.**

In his paper, “The motivation of Gameplay”, Mark Prensky (Prensky, 2002) refers to the relationship between fun and learning, and concludes that, based in part on the works of (Rose & Nicholl, 1998) and (Bisson & Luckner, 1996), fun in the learning process creates relaxation and motivation. He continues to define the benefits of the above consequences as follows: “Relaxation enables learners to take things in more easily; motivation enables them to put forth effort without resentment”.

* + 1. Motivation

Motivation has long been considered an important step in learning (Smith, 2008; Paras & Bizzocchi, 2005; Prensky, 2002). It has always been one of the teacher’s primary roles to provide students with the motivation required to stick with the learning process to the end. However, motivating students has also been one of the biggest problems in education over the last half century or so (Malone T. W., 1980; Prensky, 2002).

Without compelling content, students are left to find the necessary motivation themselves. Excluding some rare cases in which students are actually connecting with the learnt material, Prensky (Prensky, 2002) states the following as possible student motives: “students’ motives for learning are a mixture of intrinsic goals and extrinsic rewards, combined with psychological factors such as fear and the need to please”.

Malone (Malone T. W., 1980) differentiates two kinds of motivation: intrinsic and extrinsic; “In general, an activity is said to be intrinsically motivated if there is no obvious reward associated with the activity. Conversely, an activity is said to be extrinsically motivated if engaging in the activity leads to some external reward like food, money, or social reinforcement. Generally, an external reward is dispensed by a human or mechanical agent in a way that is not “naturally” a part of the rewarded activity”.

Malone emphasizes that intrinsic motivation should not be viewed as the absence of rewards, but as “a need for competence and self-, as a search for an optimal amount of psychological incongruity, or the “experience of flow””. At last, Malone states that intrinsically motivated students are likely to spend more time and effort learning and feel better about what they learn.

* + 1. Flow

The literature commonly references the Flow Theory by Csikszentmihalyi (Csikszentmihalyi, 1990) within the concept of motivation and learning, and can be regarded as a method for understanding and implementing motivation (Paras & Bizzocchi, 2005). A state of flow can be described as an “Optimal experience, where we feel a sense of exhilaration, a deep sense of enjoyment that is long cherished, does not come through passive, receptive, relaxing times.” (Csikszentmihalyi, 1990). Csikszentmihalyi describes that a state of flow in an activity can be achieved through having clear goals, achievable challenges and accurate feedback (McGinnis, Bustard, Black, & Charles, 2008). This is a psychological take that can provide a rational explanation as to why and how games may benefit a learner, since it is generally considered that most games put a person in a state of flow.

* + 1. Play

Paras and Bizzochi (Paras & Bizzocchi, 2005) state that play, which is the state a person is at when playing games, produces a state a flow. They assert that **“**Just like learning environments shouldn’t restrict the learner’s ability to more freely construct knowledge, the game environment should not restrict the player’s cognitive process, but rather allow the game player to freely make choices that help to reach an end goal**”.**

According to Paras and Bizzocchi (Paras & Bizzocchi, 2005) “**Games foster play, which produces a state of flow, which increases motivation, which supports the learning process**”.

1. Computer Game Enhanced E-Learning Systems

Based on the above, games are inherently motivating for players. We would like then to introduce the same elements that make games motivating, to a learning environment.

* 1. Principles of Instructional Game Design

What is it then that makes games so captivating? Malone (Malone T. W., 1980) defines three major categories:

*Challenge*

**Many players enjoy playing games because they provide a challenge**. In single-player home games, where social or bragging rights motivations are less of an issue, it is a primary source of motivation: “When a person faces a challenge and then overcomes it, that person has learned something. It does not matter if that challenge is in a math textbook or in a computer game. Challenging games can be learning experiences. Players will learn from games, even if that learning is limited to the context of the game, such as how to navigate through the forest, survive a particularly hairy battle, or convince the duke that their intentions with his daughter are honorable.” (Rouse, 2005)

Learners have indicated that unchallenging material fails to stimulate them, making the experience unattractive and eventually discouraging progression (McGinnis, Bustard, Black, & Charles, 2008).

Providing players (and learners) with a challenge seems to be the most appealing motivating aspect for playing (and thus learning). Specifically, providing a clear, reachable and appealing goal to a game (Malone T. W., 1980; Rouse, 2005; McGinnis, Bustard, Black, & Charles, 2008; Csikszentmihalyi, 1990).

*Fantasy*

Malone defines a fantasy inducing environment as “one that evokes mental images of things not present to the senses or within the actual experience of the person involved”, and states that “fantasies can make instructional environments more interesting and more educational” (Malone T. W., 1980).

*Curiosity*

One of the main motivating forces that propel players through a game is the desire to explore new things (Rouse, 2005). Malone describes curiosity in the context of learning as “the motivation to learn, independent of any goal seeking or fantasy-fulfillment” (Malone T. W., 1980). In general, the learning environment should be one where “the learner knows enough to have expectations about what will happen, but where these expectations are sometimes unmet.

We can distinguish between the following aspects of curiosity (Malone T. W., 1980):

1. Sensory curiosity – attracting player attention through changes in patterns of audio or visual effects.
2. Cognitive curiosity – the desire to improve one’s knowledge structure.

In his book, “Game Design, Theory and Practice”, Richard Rouse III (Rouse, 2005) lists a number of reasons players play games. We may augment the above three principles with the following three:

*Socialization*

The origin of games is tied to a social experience, and that is something all game designers must remember. This communal component is central to their appeal: “For most people, **the primary reason they play games is to have a social experience** with their friends or family.” (Rouse, 2005)

*Bragging Rights*

“**When** **players are victorious at a challenging game, they realize they can do something well, probably better than most people, which makes them feel better about themselves**.” (Rouse, 2005)

*Interaction*

The one thing the art of games can do better than any other art form is providing an interactive experience. **Players play because they want to interact** (Rouse, 2005).

* 1. Heuristics for the Design and Evaluation of Instructional Computer Games

In her paper, “Heuristics and usability guidelines for the creation and evaluation of fun in video games” (Federoff, 2002), Melissa Federoff compiles a list of heuristics used to create fun in games.   
The list is based mostly on available literature, and augmented with a few heuristics derived from her case study. We shall use some of the entries in the list to evaluate existing solutions, and to provide us with guidelines in creating and evaluating our own solution. Table 1 presents some of the key principles from that list.

|  |  |
| --- | --- |
| **Element** | **Guideline** |
| Game Interface | A player should always be able to identify their score/status in the game |
| Game Interface | Use sound to provide meaningful feedback |
| Game Interface | Provide means for error prevention and recovery through the use of warning messages |
| Game Interface | Players should be able to save games in different states |
| Game Mechanics | Feedback should be given immediately to display user control |
| Game Play | There should be a clear overriding goal of the game presented early |
| Game Play | There should be variable difficulty level |
| Game Play | There should be multiple goals on each level |
| Game Play | Game play should be balanced so that there is no definite way to win |
| Game Play | The game should give hints, but not too many |
| Game Play | The game should give rewards |
| Game Play | There must not be any single optimal winning strategy |
| Game Play | Design for multiple paths through the game |

Table 1

* 1. Designing Video Games for Foreign Language Learning

We shall complement the above list with a number of key principles that relate to the current work of English language learning, taken from “10 key principles for designing video games for foreign language learning” (Purushotma, Thorne, & Wheatley, 2009):

1. All elements of the game, particularly communication and input mechanisms, should have a playful spirit to them.

“Having students type out full sentence responses is, naturally, too slow and cumbersome for use in a video game. Thus, most games simply provide students with menus offering choices of preconstructed sentences to choose from by clicking on them with their mouse. Yet, this removes the process of constructing a sentence from the student, making it questionable what, if any, learning of language forms is taking place.”

1. Metalinguistic descriptions and terminology should be presented through optional supporting material, not as part of the core gameplay.

“It is generally not necessary — or desirable — to place such metalinguistic information as central feature of the game; nor should the game require players to name the grammatical categories and other metalinguistic concepts in order to make progress in the game, so long as they are able to use them successfully. In fact, many of the bestselling language learning titles (like Pimsleur and Rosetta Stone) take the extreme position of forgoing any mention of metalinguistic terms at all.”

1. Evaluating Existing Solutions

This section serves two purposes:

1. To conclude whether or not the option of selecting an off the shelf product that answers all of the most important requirements is valid.
2. To verify the uniqueness of our proposed solution with regards to English language structure learning.

To achieve the first goal, we evaluated different kinds of quiz engines and learning management systems. At this end, all solutions evaluated lacked gameplay elements.

To achieve the second goal, we evaluated various English learning games. While all solutions evaluated were targeted to a younger audience, almost all answered all or most of the criterions used to evaluate good games, as noted by (Malone T. W., 1980).

Table 2 presents a Comparison of current solutions and our proposed solution, based on baseline set of design principles and requirements, partially taken from Table 1.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Guideline** | **English Learning Games** | | | | | **LLS** | **LMS** | | Prodigy |
| [Alien Language](http://crossland.co.uk/alienlanguage.co.uk/alienlanguage/index.htm) | [The Magic Key](http://www.bbc.co.uk/schools/magickey/adventures/index.shtml) | [Mingoville](http://www.mingoville.com) | [Monkey Puzzles](http://www.cambridgeesol.org/exam-preparation/games/monkey-puzzles.html) | [Academy Island](http://www.cambridgeesol.org/exam-preparation/games/academy-island-game.html) | [Rosetta Stone](http://www.rosettastone.com/) | [Quizlet](http://quizlet.com/) | [Moodle](http://moodle.com/) |
| In game score/status | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗴 | ✓ |
| Audio feedback | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗴 | ? |
| Error prevention | 🗴 | 🗴 | 🗴 | 🗴 | 🗴 | 🗸 | 🗴 | 🗸 | 🗸 |
| Save state | 🗴 | 🗴 | 🗸 | 🗴 | 🗸 | 🗸 | 🗴 | 🗴 | 🗸 |
| Immediate feedback | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗴 | 🗸 |
| Clear goal | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗴 | 🗴 | 🗸 |
| Variable difficulty level | 🗴 | 🗴 | 🗴 | 🗴 | 🗸 | 🗴 | 🗴 | 🗴 | 🗸 |
| Multiple level goals | 🗴 | 🗴 | 🗸 | 🗸 | 🗸 | 🗸 | 🗴 | 🗴 | 🗸 |
| More than one way to win | 🗴 | 🗴 | 🗴 | 🗴 | 🗴 | 🗴 | 🗴 | 🗴 | ? |
| Give few hints | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |
| Give rewards | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗴 | 🗴 | 🗴 | 🗸 |
| No optimal strategy | 🗴 | 🗴 | 🗴 | 🗴 | 🗴 | 🗴 | 🗴 | 🗴 | 🗴 |
| Multiple paths through game | 🗴 | 🗴 | 🗸 | 🗴 | 🗸 | 🗴 | 🗴 | 🗴 | ? |
| Socialization | 🗴 | 🗴 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 | 🗸 |
| Multiplayer | 🗴 | 🗴 | 🗸 | 🗴 | 🗴 | 🗴 | 🗸 | 🗴 | 🗴 |

Table 2

1. Technologies

This project shall be implemented as a web-based application. Web-based applications are often multitier applications that divide functionality into tiers. Usually, these tiers are located on different computers, although it is possible that they will all reside on the same computer.

The three tier architecture is the most common one:

1. Bottom Tier: also known as the Data tier. Maintains the application’s data, usually in a relational database management system.
2. Middle Tier: implements business logic. As the name implies, this tier serves as a mediator between the application’s clients and its data.
3. Top Tier: this is the application’s user interface, through which clients interact directly. Also known as the client tier.

There is a variety of technologies available at each tier. While we shall not attempt to cover all available ones (there is little time and even less available space in this paper), we shall briefly discuss the technologies most familiar to us, which are also the ones we shall be comparing. As Steve McConnoll notes in “Code Complete 2” (McConnell, 2004), “Programmers are more productive using a familiar language than an unfamiliar one”.

* 1. Bottom Tier Technologies
     1. RDBMS

A relational database stores data in data tables. To enhance modularity between the tiers, the database is usually available as a separate product, the relational database management system. The RDBMS is usually implemented as a server component, controlling access to the tables themselves. We shall be examining three popular free license databases management systems.

* + - 1. MySQL

**MySQL**, now owned by Oracle, is a multiuser, multithreaded (i.e., allows multiple connections) RDBMS server that uses SQL to interact with and manipulate data. According to (Wikipedia, 2012), it is the world’s most used open source RDBMS as of 2008.

* + - 1. Microsoft SQL Server Express

**Microsoft SQL Server Express** is a free version of Microsoft’s extremely popular SQL Server RDBMS. It is designed specifically for small scale applications. Much like the rest of Microsoft’s products, it is highly supported and very user friendly. It is however designed to work solely with Windows systems. This edition also limits the size of usable RAM and the size of the database itself.

* + - 1. Oracle Express Edition

**Oracle Express Edition** was first introduced in 2005, and is offered as a free distribution on Windows and Linux machines. Oracle’s superb database capabilities are mostly apparent in this version, but much like Microsoft’s slimmed down version, it also limits the number of usable CPUs, RAM and user data.

Table 3 compares the above technologies based on selected criteria. We have selected MySQL as our RDBMS of choice, mainly due to the full set of features and unlimited performance and storage capabilities available with the free edition. Also, due to its popularity, it is highly supported throughout the community.

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Oracle 11g XE** | **MS SQL Server Express** | **MySQL Community Edition** |
| CPU usage count | 1 | 1 | Unlimited |
| License | Free | Free | Free |
| Support | 🗸 | 🗸 | 🗴 |
| GUI Administration | 🗸 | 🗸 | Workbench |
| Procedures | 🗸 | 🗸 | 🗸 |
| Functions | 🗸 | 🗸 | 🗸 |
| Triggers | 🗸 | 🗸 | 🗸 |
| Windows | 🗸 | 🗸 | 🗸 |
| Mac OSX | 🗸 | 🗴 | 🗸 |
| Linux | 🗸 | 🗴 | 🗸 |
| Unix | 🗸 | 🗴 | 🗸 |

Table 3

* 1. Middle Tier Technologies
     1. Programming Language (referenced from (McConnell, 2004))

We are admittedly partial towards object oriented programming languages. Objects are reusable software components. Almost every noun can be represented as a software object in terms of attributes and behaviors. Programming in objects is extremely expressive, and can foster modularity, allowing the components to change and evolve with additional requirements and design constraints, with minimal effect on each other. We shall be examining two very popular languages for server side development.

* + - 1. Java

**Java** is an object oriented language with syntax similar to C and C++ that was developed by Sun Microsystems, Inc. Java was designed to run on any platform by converting Java source code to byte code, which is then run in each platform within an environment known as a virtual machine. Java is in widespread use for programming Web applications.

* + - 1. C#

**C#** (pronounced C sharp) is a general-purpose, object-oriented language and programming environment developed by Microsoft with syntax similar to C, C++ and Java, and it provides extensive tools that aid development on Microsoft platforms.

C# actually has a lot in favor development wise, in our opinion. Perhaps it is because we are more familiar with it. But one cannot deny the simplicity of using a single IDE (Integrated development environment) that supplies every available functionality required, out of the box. Compare this with Java, that due to its open source nature, has multiple IDEs available from different vendors, each supporting a different set of plugins that enhance and ease the development of the different components, such as GUI and Web components.

However, we have chosen Java as our programming language of choice, mainly because we are less familiar with it, but also because it is more commonly used in web based projects.

* + 1. Application Servers

A web server is a specialized software that responds to client requests by providing it with resources. While the business logic is implemented using one of the programming languages reviewed above, a web server is required to for the client to access and invoke this functionality.

* + - 1. Microsoft IIS

**Microsoft Internet Information Services** is the third most popular server in the world (IIS, 2012). It is designed for use with Microsoft Windows.

* + - 1. Apache Tomcat

**Apache Tomcat** is an open source software implementation of the Java Servlet and Java Server Pages technologies. Tomcat provides a pure java HTTP web server for java code to run. Tomcat is in wide use and very popular amongst web application development. (Apache Tomcat, 2012)

* + - 1. Jetty

**Jetty** is a pure java based HTTP server and Java Servlet container. It is developed as a free open source project as part of the Eclipse Foundation. It is used in numerous large scale projects, and is rapidly gaining popularity over the past years.

Table 4 compares the above based on selected criteria. We have selected Apache Tomcat as our application server of choice, mainly due to our previous familiarity with it, and its extensive community support.

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Jetty** | **IIS** | **Tomcat** |
| License | Free | With Licensed Windows System | Free |
| Web Socket | 🗸 | 🗸 | 🗸 |
| REST | 🗸 | 🗸 | 🗸 |
| Servlet Container | 🗸 | 🗴 | 🗸 |
| Support | Medium | High | High |

Table 4

* 1. Client Tier (referenced from (Deitel, Deitel, & Deitel, 2012))

Dealing with web applications, we have a number of choices when implementing the client tier. Both programming languages reviewed at the middle tier have technologies that support running applications written in those languages at the client tier. This allows for faster programming and implementation, and reuse of logic and data structures between tiers, because a person is dealing with one language for both the client and middle tier.

However, the true benefit of web based application is HTML.

* + 1. HTML5

**HTML** is a special type of computer language called a markup language. It is designed to specify the content and structure of web pages in a portable manner.

HTML enables a developer to create content that will render appropriately across the extraordinary range of devices connected to the internet. This includes smartphones, tablets, laptops, desktops and more.

Now under development, HTML5 is the new emerging version of HTML.

HTML5 references two additional technologies:

* + - 1. JavaScript

**JavaScript** is a language that helps one build dynamic web pages and computer applications. It enables one to do the client side programming of web applications.

* + - 1. Cascading Style Sheets

**Cascading Style Sheets** (CSS) is used to specify the presentation, or style, of elements on a web page.

* + 1. jQuery

**jQuery** is currently the most popular of hundreds of JavaScript libraries. jQuery simplifies JavaScript programming by making it easier to manipulate a web page’s elements and interact with servers in a portable manner across various web browsers. It provides a library of custom graphical user interface (GUI) controls (beyond the basic GUI controls provided by HTML5) that can be used to enhance the look and feel of your web pages.

* 1. Client and Middle Tier Communication Protocol
     1. REST

**Representational state transfer** (REST) refers to an architectural style for implementing web services.

The REST protocol follow a synchronous fashion: A request is made by the client, followed by a response from the server (Pull). It is impossible for the server to initiate communication with the client, although some methods simulate such Push capabilities.

* + 1. WebSocket

**WebSocket** technology provides a full-duplex communications channel. It allows true Push capabilities, meaning that the server is free to update the client with data, as soon as new data is received without the client specifically requesting for update. It is currently being standardized by the W3C. (WebSocket, 2012)

* + 1. JSON

**JavaScript Object Notation** is an alternative to XML (Extensive Markup Language) for representing data. It is a text-based data-interchange format used to represent objects in JavaScript as collections of name/value pairs. Json notation is identical to declaring objects in JavaScript (and hence the name) meaning that creating objects from JSON is extremely fast and efficient (reading data and loading to memory).

A number of third party libraries ([GSON](http://code.google.com/p/google-gson/), [Jackson](http://jackson.codehaus.org/)) that are capable of converting JSON objects to and from Java objects, extremely simplifying the development of a Java – JavaScript paradigm project.

This project will make use of all of the above client side technologies.

* 1. Development Process (referenced from (Larman, 2007))

A software development process describes an approach to building, deploying, and possibly maintaining software.

* + 1. Iterative

In an iterative lifecycle process, development is organized into a series of short, fixed-length mini-projects called iterations; the outcome of each is a tested, integrated, and executable partial system. Each iteration includes its own requirements analysis, design, implementation, and testing activities.

* + 1. Waterfall

In a waterfall lifecycle process there is an attempt to define all or most of the requirements before programming, and often, to create a thorough design before programming.

The iterative approach is especially suitable to a project of this nature, since further the full set of requirements shall be compiled throughout the development process.

* 1. Integrated Development Environments (IDE)

To support development of any software project, although not mandatory, IDE software greatly simplifies the coding process. IDEs are designed to maximize a programmer’s productivity, for one by greatly reducing configuration requirements. (Integrated Development Environments, 2012)

* + 1. Microsoft Visual Studio

Microsoft Visual Studio is an IDE from Microsoft. It is used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web applications, and web services for all platforms supported by Microsoft Windows. A paid license is required for the full capable version. (Microsoft Visual Studio, 2012)

* + 1. Eclipse

Eclipse is a multi-language IDE comprising an extensible plug-in system. It is a free software, highly popular for development of Java applications. (Eclipse (software), 2012)

* + 1. Intellij

Intellij is a commercial Java IDE by JetBrains. Intellij requires a paid license to access the full featured version, which, from experience, supports extremely rapid and fluid coding.

Table 5 compares the above technologies based on selected criteria. We have selected Intellij IDEA as the IDE of choice, mainly due to its extensive set of coding supporting features, ready out-of-the-box.

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Visual Studio** | **Eclipse** | **Intellij IDEA** |
| Java Support | 🗴 | 🗸 | 🗸 |
| C# Support | 🗸 | 🗴 | 🗴 |
| JavaScript Support | 🗸 | Via Plugin | 🗸 |
| HTML Support | 🗸 | Via Plugin | 🗸 |
| License | Paid | Free | Paid |

Table 5

GLOSSARY

**Alien Language** – Alien language is an online Computer Assisted Learning site, designed to support the teaching of modern Foreign Languages to school children 11-14, although it can be used by learners of all ages . <http://crossland.co.uk/alienlanguage.co.uk/alienlanguage/index.htm>

**The Magic Key** – The magic key website is aimed at children aged 5-6 years and follows the TV programs in which Kiepper, Biff, Chip, Wilf, Wilma and Floppy go off on their adventures. It covers English sentence and text level work. <http://www.bbc.co.uk/schools/magickey/adventures/index.shtml>

**Mingoville** - Mingoville is an interactive English language course that caters to kids of all ages worldwide. Mingoville consists of a wide variety of materials that can be reached directly through the portal. Mingoville is a storybook world of English language learning and interaction. There are 10 missions in the course. Each mission covers a distinct theme and can be worked on independently from the other missions. <http://www.mingoville.com/en/demo.html>

**Monkey Puzzles** – Monkey Puzzles is a collection of 8 progressively unlocked mini-games designed to test knowledge as a preparation to the Cambridge English Exams. <http://www.cambridgeesol.org/exam-preparation/games/monkey-puzzles.html>

**Academy Island** – Players of Academy Island have to progress through different levels by helping an unknown alien life form use English in a range of situations such as shopping in a bakery and visiting places such as an art gallery and library. <http://www.cambridgeesol.org/exam-preparation/games/academy-island-game.html>

**Rosetta Stone** – Rosetta Stone is multiple language learning software, that uses patented, smart speech recognition tools. <http://www.rosettastone.com/>

**Quizlet** – Quizlet is an online learning tool. It is based on the concept of Sets, i.e. multiple question-answer pairs that users create and reference alike. <http://quizlet.com/>

**Moodle** – Moodle is a popular open source Learning Management Software. <http://moodle.com/>

REFERENCES

*Apache Tomcat*. (2012, 12 15). Retrieved 12 18, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Apache\_Tomcat

*Eclipse (software)*. (2012, 12 17). Retrieved 12 18, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Eclipse\_(software)

*IIS*. (2012, 12 16). Retrieved 12 18, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Internet\_Information\_Services

*Integrated Development Environments*. (2012, 12 15). Retrieved 12 18, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Integrated\_development\_environment

*Microsoft Visual Studio*. (2012, 12 17). Retrieved 12 18, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Visual\_studio

*WebSocket*. (2012, 12 17). Retrieved 12 18, 2012, from Wikipedia: http://en.wikipedia.org/wiki/WebSocket

Bisson, C., & Luckner, J. (1996). Fun in Learning: The Pedagogical Role of Fun in Adventure Education. *Journal of Experimental Education*, 109-110.

Clark, R. C., & Mayer, R. E. (2011). *E-learning and the science of instruction: Proven guidelines for consumers and designers of multimedia learning.* San Francisco: Pfeiffer.

Csikszentmihalyi, M. (1990). *Flow: The Psychology of Optimal Experience.* Harper and Row.

Deitel, P., Deitel, H., & Deitel, A. (2012). *Internet & World Wide Web: How to Program.* Prentice Hall.

Federoff, M. A. (2002). *Heuristics and Usability Guidelines for the Creation and Evaluation of Fun in Video Games.*

Jong, M., Shang, J., Lee, F., & Law, H. (2006). Learning Online: A Comparative Study of a Situated Game-Based Approach and a Traditional Web-Based Approach. *Technologies for E-Learning and Digital Entertainment, 3942*, 541-551.

Larman, C. (2007). *Applying Uml and Patterns.* New Jersey: Prentice Hall PTR.

Malone, T. W. (1980, August). What Makes Things Fun to Learn? A Study of Intrinsically Motivating Computer Games.

Malone, T. W. (1980). What Makes Things Fun to Learn? Heuristics for Designing Instructional Computer Games.

McConnell, S. (2004). *Code Complete, Second Edition.* Washington: Microsoft Press.

McGinnis, T., Bustard, D. W., Black, M., & Charles, D. (2008). Enhancing E-Learning Engagement using Design Patters from Computer Games. *First International Conference on Advances in Computer-Human Interaction.* Sainte Luce: IEEE Computer Society.

Paras, B., & Bizzocchi, J. (2005). Game, Motivation, and Effective Learning: An Integrated Model for Educational Game Design. *Changing Views - Worlds in Play.* DiGRA.

Pivec, M. (2007). Editorial: Play and learn: potentials of game-based learning. *British Journal of Educational Technology*, 387-393.

Prensky, M. (2002). The Motivation of Gameplay. *On The Horizon*.

Purushotma, R., Thorne, S. L., & Wheatley, J. (2009). *10 Key Principles for Designing Video Games for Foreign Language Learning.*

Rose, C., & Nicholl, M. J. (1998). *Accelerated Learning for the 21st Century.*

Rouse, R. I. (2005). *Game Design Theory & Practice.* Wordware Publishing Inc.

Sloman, M. (2001). *The E-Learning Revolution.* London: CIPD.

Smith, R. (2008). *Motivational Factors in E-Learning.*

Stannard, A. W. (1959). *Living English Structure, A Practice Book for Foreign Students.* Longmans.

Wang, H., & Sun, C.-T. (2001). Game Reward Systems: Gaming Expreiences and Social Meanings. *DiGRA.*

Wikipedia. (2012, 12 4). *MySQL*. Retrieved 12 18, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Mysql

Tomcat 7

MySQL Database

HTML5

(JavaScript, CSS3, jQuery)

Model Entities

HTTP GET

REST Resources

JSON

Hibernate ORM

Top Tier

Client

Middle Tier

Server

Bottom Tier

Database

AJAX

REST

Tomcat Managed Connection Pooling

MySQL Database

Hibernate

Tomcat 7

HTML5

(JavaScript, CSS3, jQuery)

Model Entities

HTTP GET

REST Resources

JSON

Hibernate ORM

Top Tier

Client

Middle Tier

Server

AJAX

Bottom Tier

Database

REST

Tomcat Managed Connection Pooling

REST API

/subjects

{

“subjects”: [

{

“name”: *String*,

“exercises”: {

“count”: *Number*

“href”: *URI*

}

}

]

}

/subjects/{id}

{

“name”: *String*

“exercises”: {

“count”: *Number*

“href”: *URI*

}

. . . *// any other attributes to be added in the future*

}

/subjects/{id}/exercises

{

“subject”: { “name”: *String }*

“exercises“: [

{

“title”: *String*,

“difficultyLevel”: *Number*

“questions”: {

“href”: *URI*

“count”: *Number*

}

},

. . .

]

}

/subjects/{id}/exercises/{id}

{

“title”: *String*

“questions”: {

“count”: *Number*

“href”: *URI*

}

“difficultyLevel”: {

“level”: *Number*

}

. . . *// any other attributes to be added in the future*

}

/subjects/{id}/exercises/{id}/questions

{

“subject”: { “name”: *String* }

“exercise”: { “title”: *String* }

“questions”: [

{ “href”: *URI }*,

. . .

]

}

/subjects/{id}/exercises/{id}/questions/{number}

{

“self” : { “href”: *URI* },

“html”: *String,*

“answers”: {

“href”: *URI*

},

“exercise”: { “title”: *String* },

“subject”: { “name”: *String* },

“difficultyLevel”: {

“level”: *Number*

“name”: *String*

}

}

/subjects/{id}/exercises/{id}/questions/{number}/answers

{

“subject”: { “name”: *String* },

“exercise”: { “title”: *String* },

“question”: { “number”: *Number*},

“answers”: [

“answer” : {

“text” : *String*

“index”: *Number*

“correct”: *Boolean*

“feedback”: String

},

. . .

]

}

/difficultyLevels

{

“difficultyLevels”: [

{

“level”: *Number*

“name”: *String*

},

. . .

]

}

ANALYSIS AND DESIGN

In this section we shall focus on three major phases preceding the implementation of the system.

In the first phase we gather requirements of Prodigy. It consists of two sub sections: the first presents a list of goals that users of prodigy wish to achieve of the system. The second presents the vision of the game of Prodigy. The game design captures the requirements in a story like manner, and augments the requirements with implementation specific details mainly relating to user interaction. **At this stage we shall try to capture all requirements that relate to the rules of the game, as they will emerge as we try thing out.**

In the second phase we analyze the requirements. It consists of two sub sections as well: in the first we produce the domain model artifact. The second details any architectural and technological constraints that derive from the requirements.

In the third and last phase we design the system in software terms. This phase contains 4 sub sections: the first presents the system architecture. The next three describe in limited detail the 3 tiers – the database, which includes an Entity relationship diagram, the server, in which we identify major classes and functionality, and the client, where we describe the UI components and major client side logic.

1. Requirements
   1. Functional
2. Get a list of questions by subject | exercise | difficulty level
3. Fill in blanks in a question from different multiple choices assigned to each blank.
4. View the question with selected answers in blanks before checking if the submitted answers are correct of false.
5. Move back and forward between questions.
6. No predefined order of submitting blanks.
7. Be able to receive a hint on a selected answer (whether true or false, and appropriate feedback should appear).
8. View questions that have been completed.
9. Receive textual feedback on a submitted answer (correct or false).
10. Display different multiple choices for each blank.
11. Receive a hint before submitting an answer (eliminate a dummy answer).
12. Have a playful mechanism for submitting answers.
13. Receive immediate feedback to display user control
14. Have multiple goals in each level
15. Receive clues and hints, but not too many
16. Have multiple paths through the game
17. Have freedom in selecting the learnt material
18. Be able to identify score/status in a game
19. Reward on performance
20. **Increase | decrease difficulty level of questions based on performance.**
21. **Store submitted answers to questions.**
22. **Store a user profile in system.**
    1. Business Rules
23. A specific answer to a blank may be submitted only once if it is incorrect.
24. After a correct answer has been submitted no other answer may be submitted to the same blank.
25. Analysis
    1. Identifying Cases of Use

Get a list of X question ids that match a supplied subject, exercise or difficulty level.

Get details of a question by its identifier.

Get details of all answers for a given blank in a question.

Get feedback on an answer.

Get the number of questions that match a supplied subject, exercise or difficulty level.

Get all subjects.

Get details of a subject by its identifier.

Get all exercises.

Get details of an exercise by its identifier.

We start by breaking down the cases of use from the most general ones. They are:

**UC1: Solve Exercises**

**UC2: Review Stats on Exercises**

**UC4: Manage Exercises**

* + 1. UC1: Solve Exercise

Playing a game is done after the user has selected a level. Levels are presented in the context of sections (subject-matter). We identify two sub-use-cases:

**UC1.1: Select Section**

**UC1.1.1: Select Exercise**

A game starts after a level has been selected and the start game action is invoked.

**UC1.2: Start Exercise**

A level has been completed once the last question has been solved.

**UC1.3: Complete Exercise**

Playing a level includes the following:

**UC1.2.1: Complete Question**

**UC1.2.2: Pause/Resume Game**

**UC1.2.3: Quit Game**

The first of the above three details the actual game loop involved in playing a game. The other two are self-explanatory.

* + 1. UC2: Review Stats

Every registered user is provided with statistical information about the games he completed. Stats are separated into three categories:

**UC2.1: Character Overview**

**UC2.1: Section Overview**

**UC2.2: Level Overview**

The first provides a general overview of the current player. Such stats as total XP earned and other player attributes.

The second and third provide similar details in the scope of individual sections and levels, respectively.

1. UC1: **Get Section Details**. Includes the following information:
   1. Name
   2. Id
   3. Number of users currently playing any level in this section
   4. Number of users who have played at least one level in this section (visitors)
2. UC2: **Get Level Details**. Includes the following information:
   1. Name
   2. Id
   3. Number of users currently playing this level
   4. Number of users who have ever played this level
   5. Average score per game out of all players
   6. Top five game scores and the names of the users who achieved them (top five players)
   7. Number of questions in the level (?)
   8. Difficulty level
   9. For each level, if completed, show details of last game played.
      1. Date played
      2. Total time of game
      3. Top score of active user
      4. Level goal reached
3. UC3: **Play Game**
   1. Solve questions by placing choices in blanks
   2. Put multiple choices by order in a blank
   3. Remove a choice from a blank
   4. Show a single choice at a time, and allow scrolling between choices
   5. Show all choices in question at once
   6. Feedback on every choice selected, true or false
   7. Get hint
   8. Show current score in game at all times
   9. Show indication on how far the user is from completing a question, and the level in general
   10. View all questions currently complete and the score received at each
4. UC4: **Review Own Achievements**
   1. Top score in each level played
   2. Total time spent in playing levels from a specific subject
   3. The levels completed at each subject

* 1. Fully Dressed Use Cases

Use Case: Complete Level

Actor: Player

Interests:

* Player: wants fast progression from one question to the next.

Preconditions: Selected level is unlocked.

Success Guarantee: Last question has been completed. Game state stored.

Main Success Scenario:

1. System loads first question – text is shown with indications of missing answers. Answers are loaded and presented to player.
2. The location of the first missing answer in the text is highlighted.
3. Player clicks an answer.
4. Highlighted missing answer is filled with the text from the selected answer and next missing answer is highlighted.
5. Player repeats 2-4 until all missing answers are filled. When last missing answer is filled, the system evaluates the solution.
6. System finds a matching solution to the one submitted by the user, and determines the next question to appear based on that solution.
7. Game state is updated by the game rules.
8. Player completes last question in game.
9. Game state is updated and stored.
10. System presents user with an overview of the completed level.
11. The user can choose to replay the level, continue to the next one or quit back to level selection.

Extensions:

1. At any time, player can pause the game:
2. System stops game timer.
3. The question and answers are hidden.
4. User selects one of the following options:
   1. Resume: resumes the paused game. Timer is resumed.
   2. Quit: quits from the game back to level selection.
5. Play may reorder filled answers within a blank:
6. Player drags a filled answer to a location within the same blank.
7. Player drops answer at a new location.
8. Filled an empty answers within the same blank are reordered.
9. Player may clear a filled answer:
10. Player double clicks the filled answer.
11. Answer is removed and placed back at the available answers.
12. A missing answer placeholder is placed at that location.
13. Player may submit a false solution:
14. System highlights the answers that are false within the context of the matching predefined solution.
15. System presents player with the tip (if one is assigned to the question).
16. The next default question is loaded.

Use Case: Start Level

Actor: Player

Preconditions: Player logged.

Success Guarantee: Game starts with selected level

Main Success Scenario:

1. User enters the level selection screen.
2. System loads a list of the names of all sections available to user.
3. The system automatically selects the last section from which a level was played or recently unlocked.
4. A section overview is presented to the user.
5. System loads a list of level names that belong to the selected section.
6. The last unlocked level is automatically selected by the system.
7. A level overview is presented to the user.
8. User clicks the start button.
9. System loads game with selected level.

Extensions:

1. The user may select a different section than the one automatically selected by system:
2. User selects a different section.
3. In case no level from the selected section is available:
   1. A message is presented to user.
   2. Alternate solutions:
      1. Don’t allow user to select such a section.
      2. Don’t present such a section to user.
4. Else operations 4-5 are performed.
5. The user may select a different level than the one automatically selected by system:
6. User selects a different level within the section
7. In case the level is unlocked:
   1. Show unlock condition to user
8. Else operations 7-8 are performed

Use Case: Complete Question

Actor: Player

Extends: Complete Level

Precondition: User is in mid game. Question loaded and answers presented to user. There is at least one unfilled answer.

Success Guarantee: next question is loaded. In case this was the last question, level is completed.

Main Success Scenario:

1. System selects first blank in question
2. User selects an answer.
3. Blank is filled with text of selected answer.
4. User repeats 1-5 until the last blank is filled.
5. System evaluates the solution of the player.
6. System updates game state and UI.
7. System updates current game state session at server.
8. System loads next question.

Extensions:

1. (7) User submitted a solution with mistakes. In this case the following scenario occurs:
2. System highlights the first mistake made in the question.
3. User selects an answer.
4. If answer is correct, system continues to the next mistake in question.

Else, // TODO: define

* 1. Domain Model
  2. Implementation Constraints

1. Design
   1. Architecture

The requirements impose a time constraint regarding validation of solution and progression through the different levels.

Top level design:

* Client – UI engine and most of the business rules (game logic).
* Server – Session management, partial business rules (multiplayer gaming in mind?)
* Database – Persistent data

At the current time, we choose to have most of the game engine implemented at client side. This includes the evaluation of solutions. The state of the game may still be sent to server for session management, and may also be validated, but the client state will vary independent of the state resulted from server. This is to provide a fast interactive experience.

The server provides the session management capabilities, which include the current state of the game. The game state will be preserved for the duration of the session (either at server or database).

The client module is divided in three major sub-modules: Section selection screen, Level selection screen and the Game. All sub modules communicate with the same server application

* 1. Database Module
     1. Schema Design
  2. Server Module
     1. Major Classes and Functions
  3. Client Module
     1. Screens
     2. Major Functions

GAME DESIGN

* 1. Game Design

A player in a game has a score, combo and health attributes associated with him. When a game starts, the score and combo are reset to zero, and the game timer starts ticking. The player’s health is at a default initial capacity of 100%. Aside from the game time, the other elements are always visible to the user.

The first challenge of the level appears. Completing a challenge is not limited by time.

Blanks can hold any number of choices, and a choice may be injected at any index within the choices contained in a blank. When a choice is placed at a blank, the other choices are retain their order if it is inserted between them.

Every challenge has one or more solutions. A solution to a challenge is a set of zero or more unique mappings of a choice to a blank, and its order within it (a tuple of <choice, blank, index>). The size of a solution is the number of such unique coordinates. To simplify matters, if a challenge has more than one solution, they must all be of the same size.

A challenge carries with it its multiple choices that a user may fill blanks with. The choices of a challenge are those that its solutions define, and optionally more dummy choices that do not appear in any of the solutions.

Choices appear below the current challenge in a separate element. We refer to it as the choice box. The choice box can be in one of two states: in the first state, a single choice is visible to the student. If the user wishes to scroll within the available choices in this state, he drags the current visible choice to special locations within the choice box to show the next or previous choice available. In the second state all choices are visible. The default state is the first. To switch to the second state, the user drags the current choice to another special location within the choice box to populate it with all available choices. To revert back to state one, the user drags any choice available in the choice box to a special location that minimizes the available choices and presents the dragged choice as the active one.

The choice box indicates to the user how many more choices he may take from it, before he has enough to complete a solution. This is basically the size of the solution, and is decreased with every choice removed from it and incremented when a choice is added to it. We refer to this attribute as the goal distance. When a challenge is loaded, goal distance is initialized to the size of the solutions.

The user is able to fill a blank in several ways: in the first, he drags a choice from the choice box to a blank in the challenge. When this occurs, two things happen: the goal distance is decremented by one, and the choice is added to the blank at the location where the user dropped it. In the second, he drags a between blanks. In this case, the choice being dragged is removed from its original blank and placed at the destination blank, in the location inserted within it.

The order in which challenges appear, are based on the solution submitted by the student. If the user submitted a solution that the challenge does not have, a default challenge should appear next, as specified by the level designer.

Once the goal distance is zero, i.e. the total number of choices present in blanks within the challenge has been reached and no more choices may be taken from the choice box. One can note that this happens when the last choice is placed at some blank. When this occurs, the following happens:

First, the solution is evaluated. If it is correct, the choices in the blanks turn green. The score is incremented by 100 points. The combo is incremented by one. The challenge folds to a single line that presents only the score received at that challenge. If the solution is false, a timer starts counting down 7 seconds, by the end of which any partial correct solution is presented by greening out correct choices. During this time, with every move the player makes a condition is evaluated: if the choices placed are all part of a single solution, they turn light green, to indicate the user that only their locations are false. Else, they turn orange. In this case, the user may remove choices from the challenge, which in turn increase the goal distance and he is able to take new choices from it (note that from here the player may again reach the state where all choices in the challenge belong to a single solution but ordered in the wrong order).

At the end of the 7 seconds granted to correct a solution, the next challenge is loaded. If no correct solution was found, the following evaluation occurs: If the solution submitted was of state 1 (correct choices, wrong order), only the combo is lost (reset to zero) and no damage is taken by the player. If the solution submitted is of state 2 (wrong choices), the combo is lost, and the player’s hitpoints are decreased by the number of

**Game Goals**

Principle: Provide an obvious goal. (Malone T. W., 1980)

The goal at each level is to complete a number of questions without losing all hit points (without “dying”).

Principle: Players must be able to tell whether they are getting closer to the goal. (Malone T. W., 1980)

The amount of questions completed and those remaining are always visible to the player. The player is indicated as he gets closer to the goal, and specifically when at a point where he is about to accomplish the goal (the last question for example).

**Uncertain Outcome**

Principle: Variable difficulty level (Malone T. W., 1980)

Levels in a section are tagged by a difficulty level. There are three mutually exclusive difficulty levels:

1. Elementary
2. Intermediate
3. Advanced

We consider two ways of determining the difficulty levels:

1. Automatically by the system (Malone T. W., 1980). Depends on the score the user received at the section. Once a threshold is exceeded, a new difficulty level is available, which introduces new levels at that difficulty level.
2. Chosen by the player (Malone T. W., 1980). Two alternatives are considered
   1. The player may select the difficulty level he wishes to play the section at. This case is mostly relevant if the progression through difficulty levels must be done linearly (first accomplish section at Elementary, then at Intermediate, etc.).
   2. The following diagram illustrates the second approach (based on Game Convexities as described in (McGinnis, Bustard, Black, & Charles, 2008)).

All levels are ordered linearly and tagged with one or more difficulty levels. The section may be completed at each difficulty level that appears in at least one node. To get access to higher level material, the player must first complete all levels at the prior difficulty level. We note however that access to more difficult material may be productive and challenging to the player, and this is something we should consider. If this is the case, accomplishing a previous node may unlock all content at one difficulty level higher until the next node with the current difficulty level is met.

Another issue regards the following: should a node at some difficulty level be unlocked only when previous same difficulty level nodes have been accomplished?

Example:

Starting difficulty level = Elementary

Entry node = level 1

Exit node = level 6

Complete level 1

Unlock level 2.

If ( Level 2 )

Complete Level 2

Complete Level 3

Unlock level 5.

If ( level 5)

Complete level 5

Complete level 6

End section.

Starting difficulty level = Intermediate

Entry node = level 2

Exit node = level 5

If (level 2 incomplete)

Complete level 2

If (level 5 incomplete)

Complete level 5

End section

Level 1

Elementary

Level 3

Elementary

Level 5

Intermediate

Level 6

Elementary

Level 4

Advanced

Level 2

Intermediate

Level 7

Advanced

Principle: Multiple Level Goals (Malone T. W., 1980).

The following table illustrates the concept of level progression and how we derive the level goals.

|  |  |  |
| --- | --- | --- |
| Question Order of Appearance | Points Needed to Reach Question | Power |
|  |  |  |
|  |  |  |
| .  .  . | | |
|  |  |  |

At any stage of appearance, the player is required to accumulate a number of points in order to progress to the next question (). Every is lower bounded by the previous question’s value, and upper bounded by the sum of the previous question’s value, and the total amount of points the question is worth. For consistency’s sake: .

We can now determine a level goal at the number of question to reach. In case we use three level goals:

Level Goal 1 – Reach Question 15 (complete 14 questions)

Level Goal 2 – Reach Question 23 (complete 22 questions)

Level Goal 3 – Complete All

Special attention should be given the information available to the player when using this kind of level design. In particular, the information relating to how the user is doing, what he needs to do next, where he is located relative to a level goal, and what moves will result in him being unable to accomplish a level goal.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Q | Hit Points | Hit point Threshold To Reach Question | Min Hit Points to Pass | Accum. Hit Points | Hit points per choice screen | Choice screens |
| 1 | 5 | 0 | 4 | 5 | 1, 1, 2, 1 | 4 |
| 2 | 11 | 4 | 9 | 16 | 2, 2, 2, 2, 2, 1 | 6 |
| 3 | 8 | 13 | 8 | 24 | 1, 3, 1, 1, 1, 1 | 6 |
| 4 | 10 | 21 | 7 | 34 | 2, 4, 2, 1, 1 | 5 |

Principle: Players must be able to know how they are doing at each moment and how close they are getting to a goal.

**Game Play**

Questions are locked and disqualified if too many mistakes have been made. We need a method for calculating the number of mistakes that can be made at a question. Questions are traversed one at a time, progressing from a completed question to the next. The player is given a number of ‘skips’ he may use to skip a question. A skip could possibly reduce the players health / loose combo etc. (some sort of penalty).

Hit Points

Every blank in a question is assigned a non-negative. This number is called a Hit Point. Its purpose is to indicate the relative difficulty the blank.

A question’s hit points are calculated as the sum of the parts of its answer.

Reward System

* Hit Points

Hit points is an integer number in the range [0,100] that is used to keep track of the amount of damage entities may endure before being incapacitated. This occurs when the hit point level reaches zero. At one hundred, the entity is perfectly undamaged. Incapacitated Question entities are considered *complete,* while when a Character loses all of its hit points, the game ends with or without accomplishing a goal.

* Level Hit Points Multiplier

This is a non-negative Real in the range (1, INFINITY]. The multiplier is used to grant more points at more advanced levels in a section.

* Question Hit Points

There are a total of 100 points (percent) to earn at every stage. Every question is assigned an amount of hit points [0,100].

* Power Ups
* Combo
* Score

The Score keeping mechanism is fairly straight forward.

* Automatically varying in-game difficulty

**Gameplay**

State

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Attribute | Type | Description |  |  |  |  |  |
| Score | Long |  |  |  |  |  |  |
| Combo | Int |  |  |  |  |  |  |
| isComplete | Boolean |  |  |  |  |  |  |
| gameOver | Boolean |  |  |  |  |  |  |
| puzzleNumber | Int | Id of current puzzle |  |  |  |  |  |
| boardSize | Int | Number of squares in puzzle |  |  |  |  |  |
| squares | List<String> | tokens on board |  |  |  |  |  |
| lastMove | From, To (Int, Int) |  |  |  |  |  |  |

**Game Loop**

**Variables:**

– Total game score.

- The player’s hit points.

– The Number of correct choices selected so far.

– Counts the number of times the event has occurred.

– Counts the number times the event has occurred.

– Counts the number of times the event has occurred.

**Entities:**

– Integer.

– A list of choices, where

– A fixed size list of answers, where

– A fixed size list of questions, where

– A list of size, where each element is a list of choices.

**Events:**

– occurs when a choice , is submitted to an answer at a location within an answer, where . That is, the answer increases in size as more choices are submitted, and every new choice may be put either as the first element, last element, or any location within the elements that currently make up the answer.

– occurs when the submitted choice is placed at its correct location, relative to the other choices previously submitted.

– occurs when the submitted choice is part of the answer, but placed at the wrong location relative to the other choices previously submitted.

– occurs when the submitted choice is not part of the set of choices of the answer.

**Functions:**

– Returns true if , false otherwise.

PlayGame(Level, Player)

Init(Level)

**Loop:**

For from 0 to

While number of

While Player is not dead

Load next question

If no more questions are available

Break

Submit choice to answer at specified location

If submission was correct

Handle correct submission

Else handle false submission

**Multiplayer Gameplay (superset of single player mode)**

A game instance (round) happens in a single level. At least one player is required to start a game.

The players are divided into two teams, each with a maximum limit of players. The purpose of the teams is to gain control over all questions in a level. This is a team effort, which also incorporates some vs. elements.

The game ends when all questions have been completed, or when a team loses all its combined hit points.

The game starts with the teams at different challenges, and the players scattered randomly across different questions within the starting challenge.

The choices available to a question are hidden until the question is engaged. This forces the player to first focus on the question at hand, and not the available choices. It also requires a certain strategy of question engagement, meaning that players must first engage the questions they are most confident about.

Once a player engages a question, the available choices are discovered. The question is now occupied by the engaging player, meaning that no other player may submit answers to that question. Players are however able to see the answers that are submitted by others.

Hit points are lost if a player submits an incorrect answer.

Every challenge in a quest (question) is supplemented with a set of possible answers. This is the main interaction. A player clicks on an answer and receives feedback. Sound shall be used to enhance the experience, indicating a correct answer or a mistake.

Engage Question Use-case

User is at an incomplete question

User clicks the engage button.

The event details are propagated to the server. The Question is added to the client and locked from other clients.

The choices and solutions are returned.

The user is at index=0 of the answer. He clicks a choice. The following happens:

1. At client side, all the answers that contain a matching choice at current answer index are retrieved.
2. If no answers match
   1. Show choice in red.
   2. Update screen with new game state.
   3. Send event details to server.
   4. Update screen with game state from server (happens when game states differ).
3. Else one or more answers are found
   1. Show selected choice in green and add to the formed choice sequence.
   2. From the matching answers, if one is found such that: answer index == answer.size-1
      1. Complete answer submitted.
      2. Repeat 2.b – 2.d.
4. Repeat all until all answers are discovered.

Features

Glossary

MAX\_CHOICES = #maximum allowed unique choices in a question = 8

MIN\_CHOICES = #minimum allowed unique choices in a question = 1

MAX\_ANSWER\_SIZE = #maximum allowed sequence of choices that makes an answer.

MAX\_DUMMIES = #maximum allowed dummy choices in a question = MAX\_CHOICES – 1

MIN\_DUMMIES = #minimum allowed unique dummies in a question = 0

DUMMIESq = #dummy choices in question q

REAL\_CHOICESi = #correct choices that can be submitted at location i of an answer

CHOICE\_SCENARIO = an empty sequence of x choices, together with the available choices.

DUMMY\_HIT\_PROBABILITYi = #dummies / #reals

HIT\_POINTS\_QUESTION = 100

SOLUTION\_SIZE = #answers left in question (>0)

COMBO = #consecutively submitted correct choices in a question

HIT\_POINTS\_MULTIPLIER = COMBO \* some normalizing factor

HIT\_POINTS\_CORRECT\_ANSWER = HIT\_POINTS \_QUESTION / #answers in question

HIT\_POINTS\_ANSWER =

QUESTION\_STREAK = #question eliminated in a row without dying.

HIT\_POINTS\_ASSIST =

The purpose and goal of each level is to accumulate as much intelligence point as possible. This is done through completing questions. Every level carries a number that acts as a hit point multiplier. This is to create an illusion of higher difficulty

Intelligence Power

This is the game score.

Every answer in a question is assigned an amount of hit points. The number of hit points granted for submitting a correct answer is calculated as a fraction of the total number of answers in a question.

The players start with an intelligence level at zero.

When an answer is correctly submitted, the player‘s intelligence level increments by the answers hit point amount, and the question loses the same amount.

When an incorrect answer is submitted, the player loses the same amount of hit points from his health.

Combo is an intelligence multiplier that takes affect when a correct answer is submitted. It starts counting when two consecutive choices have been correctly submitted, and increments with each correctly selected choice.

It rewards players for submitting a long answer; those that are made of a number of consecutive choices.

XP (Intelligence Level)

Player is granted points after every move which results in a positive outcome. For submitting a correct answer, player is granted an amount of points equal (or multiplied by some constant factor) to the weight of the missing piece. For the correct use of any item, the player is rewarded a constant amount of bonus points. Optionally, each item may have its own XP scheme, where the points relevant to the used item are increased, thus allowing upgrades.

Elimination

The ability to eliminate pieces that are irrelevant to the current challenge. This is done by first toggling the eliminate button on (or two finger touch if touch is implemented), and then clicking on an answer to eliminate. Once an answer is eliminated it does not appear as an available answer in the challenge any more - the shield stripe is shrunk. No feedback is received when eliminating an answer but additional points are granted at challenge end for correct use of ability.

Elimination is required to start a combo (?)

Submit

Submitting an answer is done by clicking on a shield stripe. When a correct answer is submitted, the UI indicates it to the user – the shield stripe is shrunk, combo increases if possible, points are rewarded and a written feedback is shown ,such as “good” or “great”.

Combo (Correct Answer Streak)

Combo is an intelligence multiplier that takes affect when a correct answer is submitted. It starts counting when two consecutive choices have been correctly submitted, and increments with each correctly selected choice.

It rewards players for submitting a long answer; those that are made of a number of consecutive choices.

Answer Streak

Shield Power

At the end of each challenge, the shield is charged based on the amount of correctly used stripes. The shield remains charge throughout all challenges in a level. It enables the use of Special Items.

The charge is based on the amount of expanded stripes at the beginning of each challenge. Every shrunk stripe (resulting from correct use only, either by submission or elimination) charges the shield with some amount. Combo should affect the shield charge.

Magic Items (Special Abilities)

The player can use special items (e.g. health or help) where each use of an item results in either a positive or negative outcome. An item must first be charged before it can be used (this can be achieved by either the combo mechanism, or using some sort of “fuel” to charge items up before or during gameplay.).

Magic Items are charged using the shield power.

// BEGIN IRRELEVANT

Once an item is selected and charged, the outcome depends on the result of the next answer submission. If the submitted answer was correct, the positive outcome of the item is in effect. If the answer was incorrect, the negative outcome is in effect.

// END IRRELEVANT

Some of the items are: (perhaps divide to different characters)

1. Heal: Increases player health by some amount. Very hard to charge.
2. Time Extension: Stops clock count for X seconds, where X is determined as a constant fraction of the initial clock time. Relevant only for timed quests.
3. Undo: provides undo operations of the last submitted piece. Cannot be used after last piece is submitted.
4. Armor – protects player from health decrease if wrong answer is submitted.
5. Wild Card – a “Joker” answer. Always correct, but the real answer is not revealed.
6. Automatic Elimination: automatically eliminates one (or possibly more) piece that is irrelevant to the current missing piece.
7. Reveal – Marks all correct choices of a solution in sequence. Starts with the first choice, if the player clicks on it, the next correct choice is marked. Currently the most powerful item.

Achievement Levels (Intelligence)

An achievement level is a condition that is evaluated during a quest, which concerns either the percentage of challenges to conquer, or that of pieces to be submitted correctly. There are three (number should not be hardcoded) achievement levels in which a character may complete a puzzle. A quest master must determine values for all three conditions when creating a quest.

**A quest is completed if during play, at least one of the achievement levels is reached.**

For example:

1. 60% = Easy
2. 80% = Normal
3. 100%+ = Hard

Additional constraints may be imposed in case of timed quests.

Solution Granting

Once a quest has been completed, the character is granted the correct answers (solution) which were submitted by him. Once a solution is in possession, it is available for review purposes at the library (a place where characters may meet, and review passed quests and their solutions).

World Progression

The world is an ordered list of nodes. Characters progress from one node to the next after accomplishing all assignments at a node. A node contains one or more quests, and assignments impose conditions on how to accomplish quests.

Solution Ranking Algorithm

The algorithm acknowledges the order of answers in submitted solutions, calculates the amount of discrepancies between

Let be the current challenge, and be the set of all existing solutions in the challenge.

We define as a vector of strings constructed from the available answers in .

Let be a predefined solution to , where is an integer in the range , and as the solution submitted by the user , where .

We refer to an item in a solution vector as , where is an integer in the range .

Finally, we define to be an item with an empty string.

The goal of the algorithm is to remove items from so that eventually it shall contain a single solution closest to .

The algorithm returns an integer vector ,where , and which holds the following flags:

1. : indicating a correct answer at index j.
2. : indicating an incorrect answer at index j.

We define the following functions:

1. : retains all solutions that contain the item .
2. : retains all solutions that equal the length of the second argument.
3. :

ANALYSIS AND DESIGN

UC1: Content Creation

Actor: Admin

The following describes the requirement – the system admin. In turn, these requirements will affect the game design.

User requests a system that can present practical exercises in multiple choice formats. Each exercise may contain any kind of textual content, and each should relate to a single subject matter, from those defined in a syllabus. User specifies that an exercise reference a difficulty level, from a set of difficulty levels defined globally (i.e. Elementary, Intermediate and Advanced).

A multiple choice question presents the user with all possible choices that may be used in a question, as defined by the teacher. User requests to indicate to the student where choices should be placed within the question. A blank space indicates the student where choices should be placed. User requests that each blank be able to contain zero or more choices. That is, it should be possible for a student to chain choices together to construct larger answers within a blank.

User wants to create questions that may be solved in more than one way, thus a question may have more than one solution. A solution to a question specifies which choices appear in each blank, and the order of choices within each of the blanks.

Teacher wants to specify the order in which questions appear, based on the solution submitted by the student. If the student submitted a solution that the question does not have, a default question should appear next, as specified by the teacher.

Teacher wants the student to be able to view the choices to a question in either one of two ways: the first presents the user with a single random choice at a time. The student can then either place the choice at a blank, or request the next or previous one. The second way presents the user with all choices that are available for that question, at once.

The user differentiates between choices that are part of some solution to a question, and those that are not (within all choices that are presented to the user). Such choices are referred to as “dummies”. He wishes to provide textual feedback when using a specific dummy choice in a question.

Teacher wants to provide an assisting tip to a question, in cases where a solution has not been found after several attempts.

It is required that the student be prevented from making the same mistake more than once.

User wants to define a list of subjects and to group questions together, based on a certain subject. Every subject may have more than one group of questions.

User specifies that a future requirement may be to track answers that a user submitted to a question.

Design Concept

The requirements of the system are captured within the concept of the system.

Prodigy is an online three tier e-learning system that supports practical content in multiple choice formats.

Content is presented through subject-matters. A subject is known as a Section. It is identified by an id, and has a name (For example – “present simple”). A section also contains textual reading material in the form of HTML.

Within a section are multiple Levels. A level is an interactive exercise made of one or more multiple choice format questions. Every level is identified by an id and is tagged with a difficulty level. A Difficulty level has a name and is defined globally within Prodigy.

A multiple choice question is presented by an HTML element, and may have an assisting tip, which is text that appears to the client as defined by the engine. The HTML contains special elements identified by their index of appearance within a question. They are referred to as Blanks, and are parsed at client side and given special attributes that make it accept choices. A blank is used to indicate the user where one or more choices should be placed. A blank may contain any number of choices.

A Choice is a textual element that may be placed at a blank. It is identified by its textual content and the question to which it belongs. The choices a question has are made of those that the solution to the question contains, and if any additional choices are defined in a question, they are known as Dummies.

Every question has at least one solution. A Solution is a set of submissions. A submission is identified by a choice, a blank and the location of the choice within the blank.

Progression from a question to the next is determined by the submitted solution to the previous one. A solution may or may not define a following question. Every question has a default next question that is used in case the user submitted a solution that does not define a following question, or if the submitted solution does not exist as a possible solution to the question (usually mistakes).

User tasks:

1. Create the list of subjects.
2. Create a number of multiple choice exercises (group of questions) for each subject.
3. Assign introductory and summary text for each exercise.
4. Create questions for each exercise, with one or more answers to each question.
5. Assign a feedback text to various possible answers for a question.
6. Create and add textual theoretical background to be accessible for reference by the student at each subject / exercise.
7. Let user review questions and answers at end of exercise.
8. Let student track own progress:
   1. Average score in each completed level
   2. Number of questions completed successfully
   3. Mistakes made at each question
   4. Average number of mistakes per question (also by subject)
   5. Time spent in subjects
9. Support the following two ways of submitting answers to a question:
   1. By building a complete answer from the choices in a ordered fashion, starting from the first missing part and until the answer is complete.
   2. By selecting the missing place in the answer to which the user wishes to submit a choice to.
10. Control the progression of students through the syllabus by setting a minimal passing criterion in each level.
11. Allow student to undo a submission in certain scenarios.
12. Consider support for multiplayer gaming.
13. Mark words that appear in the question headline as Bold, Italic, and Underscored.
14. Mark words that appear in the feedback headline as Bold, Italic, and Underscored.
15. Provide answers that are made of multiple parts, where the order of the parts doesn’t matter.
16. Each level playable at different difficulty levels.
17. Define multiple levels of goals in each level:
    1. Example: By the amount of questions that must be completed / intelligence points gained.
18. Allow use of special game clues.
19. Provide constructive and creative feedback.
20. Allow creation of answers from multiple parts.
21. Allow more than one possible solution to a question.
22. Allow a different question to appear after a correct solution, and a separate one after a false solution.

Use Cases

The following list describes functional requirements of the system. These are functions invoked by the client.

1. Get Sections Details. Includes the following information:
   1. Name
   2. Id
   3. Number of users currently playing a level in this section
   4. Number of users who have played at least one level in this section (visitors)
2. Get Level Details. Includes the following information:
   1. Name
   2. Id
   3. Number of users currently playing this level
   4. Number of users who have ever played this level
   5. Average score per game in this level
   6. Top five game scores and the names of the users who achieved them (top five players)
   7. Number of questions in the level
   8. For each level, if completed, show details of last game played.
      1. Date played
      2. Total time of game
      3. Top score of active user
      4. Level goal reached
3. Play Game
   1. Solve questions by placing choices in blanks
   2. Put multiple choices by order in a blank
   3. Remove a choice from a blank
   4. Show a single choice at a time, and allow scrolling between choices
   5. Show all choices in question
   6. Feedback on every choice selected, true or false
   7. Get hints for a question
   8. Show current score in game at all times
   9. Show indication on how far the user is from completing a question, and the level in general
   10. View all questions currently complete and the score received at each
4. Review Own Achievements
   1. Top score in each level played
   2. Total time spent in playing levels from a specific subject
   3. The levels completed at each subject
   4. the subjects completed totally

UC3: Play Game

Actor: Player

User selects a level to play. The screen shows the score of the game and an indication of the number of moves made at each question. Both are initially zero.

A level starts with the first question. The blanks that need to be filled are highlighted, and the available answers are shown as well. All blanks are initially empty. The first blank of the question receives focus.

User fills blank with an answer. The answer is added to the blank. If the blank was not empty, the user orders the new answer as he pleases.

Player starts a new game. Data is loaded and state is initiated: score, combo and power are reset to zero.

1. The next question is loaded and presented:
   1. The default solution is shown.
   2. Available choices are presented to the user.
   3. Timer starts.
2. Player makes a move. There are a number of possible moves:
   1. Player moves a piece from one blank to another.

* Append answer from origin blank to destination blank, at location selected by user
  1. Player places a new answer at a blank.
* Append answer to blank, at location selected by user
  1. Player removes a piece from a blank

1. A new question state is computed with move made.

If question is complete

* + 1. Compute score, combo and power as functions of the number of moves made, the total time taken to complete the puzzle and tips used.
    2. Present feedback
    3. If level is complete (last puzzle)
       1. Present user with game stats and goal reached
       2. Present any new achievements or records broken
       3. Present user with top game stats from all users
       4. Present options to: replay level, back to level selection

Else repeat 1-3

Else question contains mistakes

1. Repeat 2-3 until correct

At any time, player may request a tip:

* Eliminate dummy: a dummy is eliminated from the choices, if one exists.
* Hint: a general hint is given as specified by the creator of the question.
* Reveal mistake: a blank that contains a wrong answer is selected and the feedback for that blank (if one exists) is presented. (method of selection unspecified. Perhaps by tracking moves made at each square and selecting the one with the most moves made, else choose random square)

At any time, player may pause game:

* Stop timer
* Present options to: resume, replay level, back to level selection, read instructions, read introduction.

Select Choice

The basic logic of selecting a choice:

Repeat process until one of the following happens:

1. A choice sequence has been completed and matches an answer.
2. User has falsely selected choices more than the allowed number.

User clicks on a choice.

If the choice is correct

Append choice to on screen sequence

Increment combo count by 1

Increment number of correct choices by 1

Decrease 1 hit point from question

Increase user hit points by 1

Else

Append the token to the choice sequence but mark as a mistake

Reset user combo to default value

The question does not lose a hit point

If the number of hit points left in the question is lower than the amount required for achieving the minimal goal

Disengage question and lock from user

If the number of available choices is greater than 2

Show

Complete description of a single player game

A player signs in the system. He clicks the button that says ‘start’. Because the player is new, the first section of the game is shown, together with the number of levels. The latest accessible level is presented (which in this case is the first), and its details are shown:

* The introduction
* The instructions
* The difficulty level
* The goal levels that the level can be accomplished at: A goal level sets the minimal amount of intelligence points a user must accomplish in order to reach this level’s goals. For every goal that is reached, the user receives an amount of bonus points and some label, such as “GrammarKing” or perhaps something more specific to the section he is at, such as “The Count of Counting” in the “countables” section. (these may also be unlocked when playing in multiplayer mode)

The player clicks the “Play” button, which starts the game. The level data is loaded at the server and client, and once complete, the game starts at the first question.

The player now sees on screen the following:

* The headline of the question
* The number of intelligence points accumulated in the level so far (starting at zero)
* The

Server Side Development

(*inspired by sourceforge,net/apps/trac/svbg/wiki/Server-side%20Development*)

Design Rationale

High Priority

* There should be no UI necessary to test and develop server-side code.
* The server implements all game logic. Client inputs should be validated.
* The server should know the expected task for each client at each point.
  + Which player’s turn is it? reject input from other users (multiplayer)
  + Which task is expected by that player? Reject invalid input.
  + Provide end-user compatible error messages.

Lower Priority

* The server should know the data needs to be sent to a client. The data should be minimized, e.g. by sending incremental data.
* It should be possible to replay a complete game and/or have an overview of what happened in which turn after the game is finished.
* The server should minimize its own state.
* The server should be able to restore its state form persistent storage.
* The game logic should be implemented in a readable, extensible way. A state machine? Rules?

Nice To Have

* Undo/Redo functionality.
* An overview of how much time each player requires.

Client Server Communication Protocol

Single Player

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Initiator | Key | Value | Destination Process | Result | Notes |
| Client | Play Game | Level Number | Load data, send data to client, start game when client signals that data is loaded and ready |  | Request to start a new game |
| **Client** | **Move To Question Number** | **Question Number** | **Update player location** | **Notify all (multiplayer)** |  |
| Client | Engage Question |  | Validate that question is not previously engaged. Assign Question as occupied by user. | Notify all  (multiplayer) |  |
| Client | Select Choice | Choice Token | Validate selected Token, place token at question answer, at the current user location | Progress location by 1 if possible |  |
| Client | Disengage Question |  | Validate that user is able to disengage. | Notify all |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Multi Player

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Initiator | Key | Value | Destination Process | Return | Notes |
| Client | Join Game | Level Number | Load data, send data to client, start game when client signals that data is loaded and ready |  | Request to start a new game |
| **Client** | **Move To Question Number** | **Question Number** | **Update player location** | **Notify all (multiplayer)** |  |
| Client | Engage Question |  | Validate that question is not previously engaged. Assign Question as occupied by user. | Notify all  (multiplayer) |  |
| Client | Select Choice | Choice Token | Validate selected Token, place token at question answer, at the current user location |  |  |
| Client | Disengage Question |  | Validate that user is able to disengage. | Notify all |  |
| Client | Use Armor |  | Validate action |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

GLOSSARY

|  |  |  |
| --- | --- | --- |
| **Term** | **Meaning** |  |
| Subject | The subject-matter of practice (e.g. Present Simple, Past Perfect etc.) |  |
| Exercise | A unit of practice that relates to a single subject. |  |
| Question | A single item in an exercise that requires solving. |  |
| Answer | A possible answer to an entire question, or part of it. |  |
| Feedback | Text that is associated with an answer and explains why the answer is correct/incorrect. |  |
| Blank | A single missing location where an answer should be placed (?) |  |
|  |  |  |
|  |  |  |