**SYNOPSIS**

**Project Title : “Snake Game”**

**Abstract :**

***Snake*** **Game** is a [video game](http://en.wikipedia.org/wiki/Video_game) that originated during the late 1970s in [arcades](http://en.wikipedia.org/wiki/Video_arcade) and has maintained popularity since then, becoming something of a classic. After it became the standard pre-loaded game on [Nokia](http://en.wikipedia.org/wiki/Nokia) phones in 1998, Snake found a massive audience.

**System requirements :**

* Microsoft Windows (7,vista,XP)
* 256 MB or higher RAM
* 800 MHz or higher Intel compatible CPU
* 200 MB free hard disk space

**Project Introduction**

The project is a “2D Snake game”. We try to make this game little bit different from the Snake game we often play cell phones**. Mostly this game is compared with NOKIA 1100 Series snake game and tried to make game slightly different from NOKIA snake game.** Games in our cell phones have **only one food** which is **kept at rest** and snake has to eat food so as to grow. But in our game we will include different types of foods having different shapes and will move in different directions with speed less then snakes speed.

Also we had included in the game the poison which will be released when **‘z’** button from the keyboard has been pressed. The poison will be in the form of fire and will help the snake to catch & eat food much easier. We had also introduced Menu’s for increasing & decreasing snake speed during play time.

We will use data files for storing our player’s names, scores and their high scores as well in **Encrypted form** using XOR encryption algorithm.

**Main features of our game are as follows**:

* **Moving foods** having different shapes, colors and speeds.
* Snake possess **poison** **in form of fire** which will move in forward direction (like bombs released by Mario).
* Allow user to **speed up and down** snake during game play as well.
* Hurdles in form of **flying predator bird** to increase difficulty level.
* **Auto save Feature** is provided so that user can load his/her saved game in future whenever required.
* **Little bit Artificial Intelligence** is provided to moving foods for prevent them being eaten by snake.
* Storing players name and score as well as their high score in encrypted form in data files by **XOR Cypher encryption algorithm.**

**Disadvantage** :

* This game supports only MS-Windows platform only.

**PROJECT PROPOSAL**

**Approach :**

In this project “SNAKE GAME” we are using the bottom-up approach**.**

**Bottom-Up Approach:**

In a bottom-up approach the individual base elements of the system are first specified in great detail. These elements are then linked together to form larger subsystems, which then in turn are linked, sometimes in many levels, until a complete top-level system is formed. This strategy often resembles a "seed" model, whereby the beginnings are small, but eventually grow in complexity and completeness. However, "organic strategies", may result in a tangle of elements and subsystems, developed in isolation, and subject to local optimization as opposed to meeting a global purpose.

**Development Environment :**

* **Operating System –** Microsoft windows (XP ,Vista ,7)
* **Front end – C++**
* **Back end – Data files & Text files**
* **Hardware** - 800 MHz or higher Intel compatible CPU

256 MB or higher RAM

200 MB free hard disk space

**C++**

**C++ (pronounced "cee plus plus**") is a [statically typed](http://en.wikipedia.org/wiki/Statically_typed), [free-form](http://en.wikipedia.org/wiki/Free-form_language), [multi-paradigm](http://en.wikipedia.org/wiki/Multi-paradigm_programming_language), [compiled](http://en.wikipedia.org/wiki/Compiled_language), general-purpose, powerful [programming language](http://en.wikipedia.org/wiki/Programming_language). It is regarded as an intermediate-level language, as it comprises a combination of both [high-level](http://en.wikipedia.org/wiki/High-level_programming_language) and [low-level](http://en.wikipedia.org/wiki/Low-level_programming_language) language features. It was **developed by**[**Bjarne Stroustrup**](http://en.wikipedia.org/wiki/Bjarne_Stroustrup)**starting in 1979 at**[Bell Labs](http://en.wikipedia.org/wiki/Bell_Labs) [object oriented](http://en.wikipedia.org/wiki/Object-oriented_programming) features such as classes to the [C programming language](http://en.wikipedia.org/wiki/C_(programming_language)). Originally named **C with Classes**, the language was renamed C++ in 1983, as a pun involving the [increment operator](http://en.wikipedia.org/wiki/Increment_operator).

C++ is one of the most popular programming languages implemented on a wide variety of hardware and operating system platforms. As an efficient compiler to native code, its **application domains including systems software,**[**application software**](http://en.wikipedia.org/wiki/Application_software)**, device drivers, embedded software, high-performance server and client applications, and entertainment software such as**[**video games**](http://en.wikipedia.org/wiki/Video_games).

**Language Features**

### Operators and operator overloading:

C++ provides more than [35 operators](http://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B), covering basic arithmetic, bit manipulation, indirection, comparisons, logical operations and others. Almost all operators can be [overloaded](http://en.wikipedia.org/wiki/Operator_overloading) for user-defined types, with a few notable exceptions such as member access (. and .\*) as well as the conditional operator.

The rich set of overloadable operators is central to using C++ as a [domain-specific language](http://en.wikipedia.org/wiki/Domain-specific_language). The overloadable operators are also an essential part of many advanced C++ programming techniques, such as [smart pointers](http://en.wikipedia.org/wiki/Smart_pointer). Overloading an operator does not change the precedence of calculations involving the operator, nor does it change the number of operands that the operator uses (any operand may however be ignored by the operator, though it will be evaluated prior to execution).

Overloaded "&&" and "||" operators lose their [short-circuit evaluation](http://en.wikipedia.org/wiki/Short-circuit_evaluation) property.

|  |  |
| --- | --- |
| **Operators that cannot be overloaded** | |
| **Operator** | **Symbol** |
| **Scope resolution operator** | :: |
| **Conditional operator** | ?: |
| **Member selection operator** | .\* |
| **"sizeof" operator** | sizeof |
| **"typeid" operator** | typeid |

### Objects:

C++ introduces [object-oriented programming](http://en.wikipedia.org/wiki/Object-oriented_programming) (OOP) features to C. It offers classes, which provide the four features commonly present in OOP (and some non-OOP) languages: [**abstraction**](http://en.wikipedia.org/wiki/Abstraction_(computer_science))**,**[**encapsulation**](http://en.wikipedia.org/wiki/Information_hiding)**,**[**inheritance**](http://en.wikipedia.org/wiki/Inheritance_(object-oriented_programming)), and polymorphism.

**Objects are instances of classes created at runtime**. One distinguishing feature of C++ classes compared to classes in other programming languages is support for deterministic [**destructors**](http://en.wikipedia.org/wiki/Destructor_(computer_science)), which in turn provide support for the [Resource Allocation is Initialization](http://en.wikipedia.org/wiki/Resource_Allocation_is_Initialization) concept.

#### Encapsulation:

[**Encapsulation**](http://en.wikipedia.org/wiki/Information_hiding)**is the hiding of information in order to ensure that data structures and operators are used as intended and to make the usage model more obvious to the developer.**

C++ provides the ability to define classes and functions as its primary encapsulation mechanisms. Within a class, members can be declared as either public, protected, or private in order to explicitly enforce encapsulation.

* A public member of the class is accessible to any function.
* A private member is accessible only to functions that are members of that class and to functions and classes explicitly granted access permission by the class ("friends").
* A protected member is accessible to members of classes that inherit from the class in addition to the class itself and any friends.

#### Inheritance:

[**Inheritance**](http://en.wikipedia.org/wiki/Inheritance_(computer_science))**allows one data type to acquire properties of other data types**. Inheritance from a [base class](http://en.wikipedia.org/wiki/Base_class) may be declared as public, protected, or private. This access specifier determines whether unrelated and derived classes can access the inherited public and protected members of the base class.

Only public inheritance corresponds to what is usually meant by "inheritance". The other two forms are much less frequently used.

**If the access specifier is omitted, a "class" inherits privately, while a "struct" inherits publicly.**

### Polymorphism:

[Polymorphism](http://en.wikipedia.org/wiki/Type_polymorphism) enables one common interface for many implementations, and for **objects to act differently under different circumstances**.

C++ supports several kinds of *static* ([compile-time](http://en.wikipedia.org/wiki/Compile-time)) and *dynamic* ([run-time](http://en.wikipedia.org/wiki/Run_time_(program_lifecycle_phase)))[polymorphisms](http://en.wikipedia.org/wiki/Polymorphism_(computer_science)" \o "Polymorphism (computer science)). Compile-time polymorphism does not allow for certain run-time decisions, while run-time polymorphism typically incurs a performance penalty.

**Data File**

A **data file** is a [computer file](http://en.wikipedia.org/wiki/Computer_file) which stores data to use by a computer [application](http://en.wikipedia.org/wiki/Application_software) or [system](http://en.wikipedia.org/wiki/System_software). Most computer programs work with [files](http://en.wikipedia.org/wiki/File). This is because files help in storing information permanently.

[Database](http://en.wikipedia.org/wiki/Database) programs create files of information. Compilers read source files and generate executable files. A file itself is a bunch of [bytes](http://en.wikipedia.org/wiki/Byte) stored on some storage device like [tape](http://en.wikipedia.org/wiki/Tape), disk, Optical etc. The **data files** are the files that store data pertaining to a specific application, for later use.

**Storage types of Data file:**

The data files can be stored in two ways:

1. Text files.
2. Binary files.

**Text files:**

A **text file** (also called ASCII files) stores information in [ASCII](http://en.wikipedia.org/wiki/ASCII) characters. A text file contains visible characters. We can see the contents of file on the monitor or edit it using any of the text editors. In text files ,each line of text is terminated,(delimited) with a special character known as [EOL (End of Line)](http://en.wikipedia.org/wiki/Newline)character. In text files some internal translations take place when this EOL character is read or written.

**Examples of text files**

* A file containing a C++ program

**Binary file**:

A **binary file** is a file that contains information in the same format in which the information is held in memory i.e. in the binary form. In binary file, there is no delimiter for a line. Also no translations occur in binary files. As a result, binary files are faster and easier for a program to read and write than the text files. As long as the file doesn't need to be read or need to be ported to a different type of system, binary files are the best way to store program information.

**Examples of binary files**

* An executable file
* An object file

**Our Understanding of Current System**

In this game, the player controls a long, thin creature, resembling a [snake](http://en.wikipedia.org/wiki/Snake), which roams around on a bordered [plane](http://en.wikipedia.org/wiki/Plane_(mathematics)), picking up food (or some other item), trying to avoid hitting its own tail or the "walls" that surround the playing area.

Each time the snake eats a piece of food, its tail grows longer, making the game increasingly difficult. The user controls the direction of the snake's head (up, down, left, or right), and the snake's body follows. The player cannot stop the snake from moving while the game is in progress, and cannot make the snake go in reverse.

However, *Snake* has had many variations since its release, depending on the game's platform. These variations involve the modification of certain rules e.g. the lethality of contact with walls.

**The first known**[**personal computer**](http://en.wikipedia.org/wiki/Personal_computer)**version of *Snake*, titled *Worm*, was programmed in 1978 by Peter Trefonas of the US on the**[**TRS-80**](http://en.wikipedia.org/wiki/TRS-80) **computer, and published by *CLOAD* magazine in the same year.**

**SYSTEM DEVELOPMENTS LYFE CYCLE**

The SDLC is a process used by a [systems analyst](http://en.wikipedia.org/wiki/Systems_analyst) to develop an [information system](http://en.wikipedia.org/wiki/Information_system), training, and user (stakeholder) ownership. Any SDLC should result in a high quality system that meets or exceeds customer expectations, reaches completion within time and cost estimates, works effectively and efficiently in the current and planned [Information Technology](http://en.wikipedia.org/wiki/Information_Technology) [infrastructure](http://en.wikipedia.org/wiki/Infrastructure), and is inexpensive to maintain and cost-effective to enhance.

**History:**

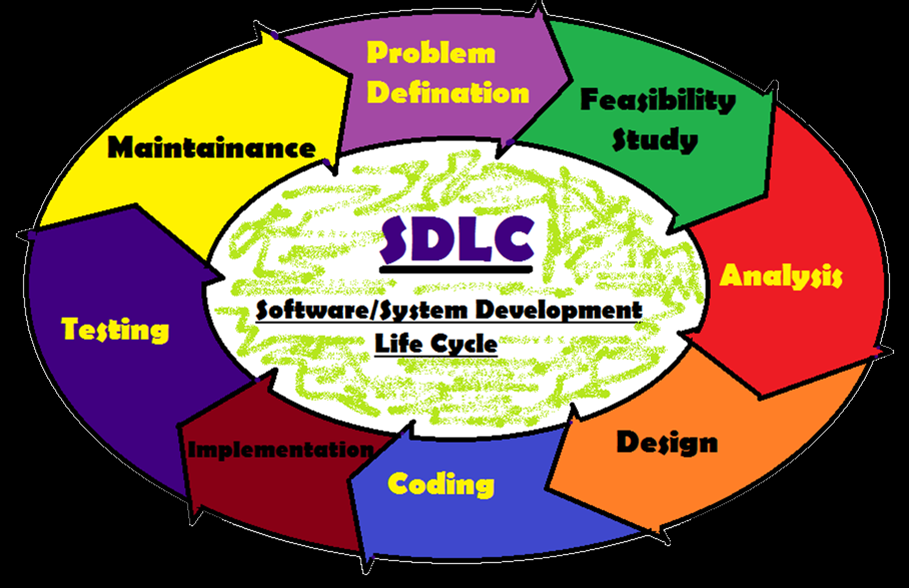
The systems life cycle (SLC) is a methodology used to describe the process for building information systems, intended to develop information systems in a very deliberate, structured and methodical way, reiterating each stage of the life cycle.

The systems development life cycle, according to Elliott & Strachan & Radford (2004), "originated in the 1960's,to develop large scale functional business systems in an age of large scale business conglomerates.

**Systems development phases:**

The System Development Life Cycle framework provides a sequence of activities for system designers and developers to follow. It consists of a set of steps or phases in which each phase of the SDLC uses the results of the previous one.

A Systems Development Life Cycle (SDLC) adheres to important phases that are essential for developers, such as [planning](http://en.wikipedia.org/wiki/Planning), [analysis](http://en.wikipedia.org/wiki/Analysis), [design](http://en.wikipedia.org/wiki/Design), and [implementation](http://en.wikipedia.org/wiki/Implementation), and are explained in the section below.



A number of system development life cycle (SDLC) models have been created The oldest of these, and the best known, is the [waterfall model](http://en.wikipedia.org/wiki/Waterfall_model) : a sequence of stages in which the output of each stage becomes the input for the next.



These stages can be characterized and divided up in different ways, including the following:

* **Preliminary analysis**: The objective of phase1 is to conduct a preliminary analysis, propose alternative solutions, describe costs and benefits and submit a preliminary plan with recommendations.

**Conduct the preliminary analysis:** In this step, you need to find out the organization's objectives and the nature and scope of the problem under study. Even if a problem refers only to a small segment of the organization itself then you need find out what the objectives of the organization itself are. Then you need to see how the problem being studied fits in with them.

**Propose alternative solutions:** In digging into the organization's objectives and specific problems, you may have already covered some solutions. Alternate proposals may come from interviewing employees, clients , suppliers, and/or consultants. You can also study what competitors are doing. With this data, you will have three choices: leave the system as is, improve it, or develop a new system. Describe the costs and benefits.

* **Systems analysis, requirements definition**: Defines project goals into defined functions and operation of the intended application. Analyzes end-user information needs.
* **Systems design**: Describes desired features and operations in detail, including screen layouts, process diagrams, pseudo code and other documentation.

During this phase, functional, support and training requirements are translated into preliminary and detailed designs. Decisions are made to address how the system will meet functional requirements.

* **Development (Coding)**: The real code is written here.

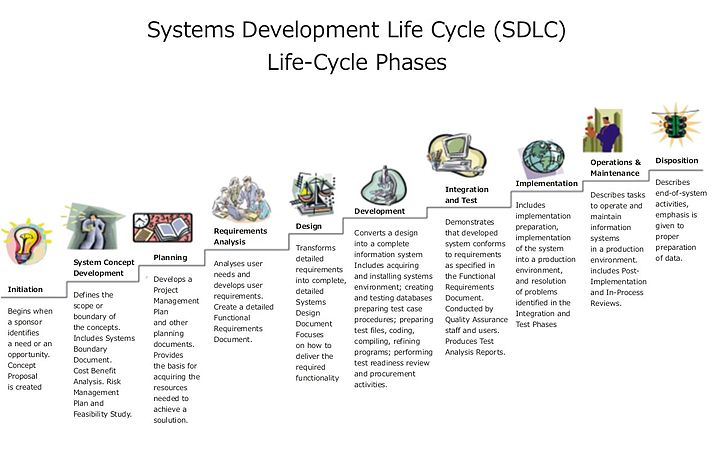
During this phase, systems are developed or acquired based on detailed design specifications . The objective is to ensure that the system functions as expected and that sponsor's requirements are satisfied.

* **Integration and testing**: Brings all the pieces together into a special testing environment, then checks for errors, bugs and interoperability.
* **Acceptance, installation, deployment**: The final stage of initial development, where the software is put into production and runs actual business.

During this phase, the new or enhanced system is installed in the production environment, users are trained, data is converted (as needed), the system is turned over to the sponsor, and business processes are evaluated.

* **Maintenance**: What happens during the rest of the software's life: changes, correction, additions, moves to a different computing platform and more. This is often the longest of the stages.

In the following example (see picture) these stage of the systems development life cycle are divided in ten steps from definition to creation and modification of IT work products:

****

The tenth phase occurs when the system is disposed of and the task performed is either eliminated or transferred to other systems. The tasks and work products for each phase are described in subsequent chapters.

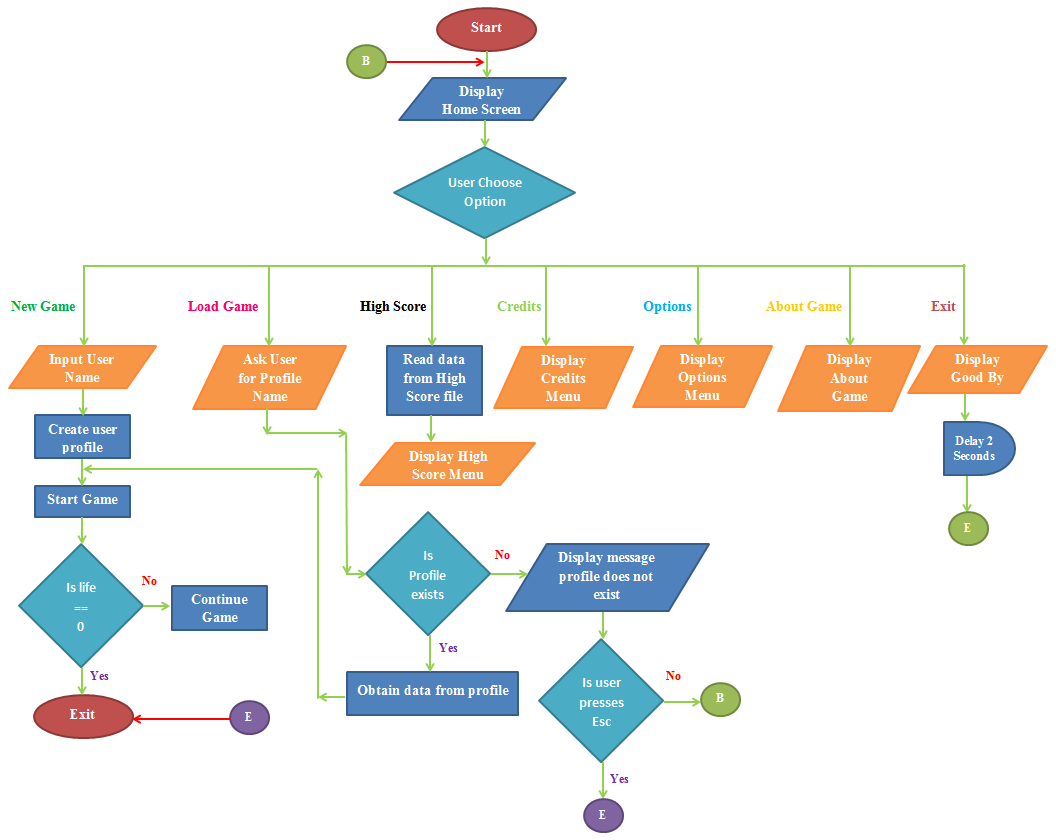
Not every project will require that the phases be sequentially executed. However, the phases are interdependent. Depending upon the size and complexity of the project, phases may be combined or may overlap.

**PROJECT DESIGN**

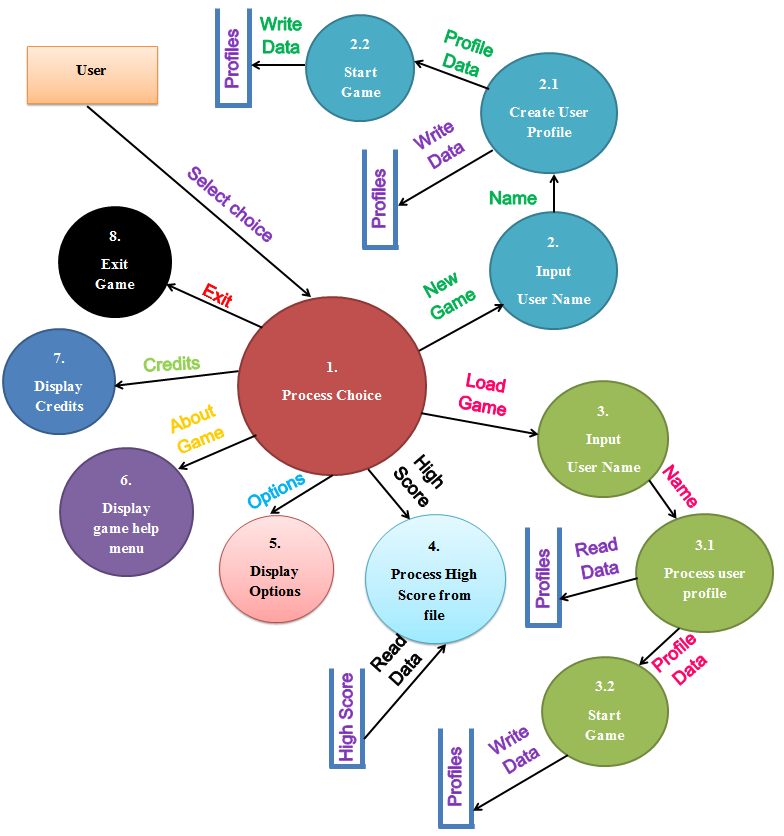
In [**Project Design**](http://en.wikipedia.org/wiki/Systems_design) the design functions and operations are described in detail, including screen layouts, game rules, **Flowchart** and **Data Flow Diagrams** (DFD’s) and other documentation. The output of this stage will describe the new system as a collection of modules or subsystems.

These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input design. The design stage takes as its initial input the requirements identified in the approved requirements document. [systems design](http://en.wikipedia.org/wiki/Systems_design) .

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts.

**Flow Chart :-**

**Data Flow Diagram :-**



**TESTING**

**Software testing** is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding [software bugs](http://en.wikipedia.org/wiki/Software_bug) (errors or other defects).

Software testing can be stated as the process of validating and verifying that a software program/application/product:

* meets the requirements that guided its design and development;
* works as expected;
* can be implemented with the same characteristics.
* satisfies the needs of stakeholders

Software testing, depending on the testing method employed, can be implemented at any time in the development process. However, most of the test effort traditionally occurs after the requirements have been defined and the coding process has been completed.

**Scope :**

A primary purpose of testing is to detect software failures so that defects may be discovered and corrected. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions.

The scope of software testing often includes examination of code as well as execution of that code in various environments and conditions as well as examining the aspects of code:

* does it do what it is supposed to do
* do what it needs to do.

[**Requirements analysis**](http://en.wikipedia.org/wiki/Requirements_analysis):

Testing should begin in the requirements phase of the [software development life cycle](http://en.wikipedia.org/wiki/Software_development_life_cycle). During the design phase, testers work with developers in determining what aspects of a design are testable and with what parameters those tests work.

* **Test planning**: [Test strategy](http://en.wikipedia.org/wiki/Test_strategy), [test plan](http://en.wikipedia.org/wiki/Test_plan), [test bed](http://en.wikipedia.org/wiki/Testbed)  creation. Since many activities will be carried out during testing, a plan is needed.
* **Test development**: Test procedures, [test scenarios](http://en.wikipedia.org/wiki/Scenario_test), [test cases](http://en.wikipedia.org/wiki/Test_case), test datasets, test scripts to use in testing software.
* **Test execution**: Testers execute the software based on the plans and test documents then report any errors found to the development team.
* **Test reporting**: Once testing is completed, testers generate metrics and make final reports on their [test effort](http://en.wikipedia.org/wiki/Test_effort) and whether or not the software tested is ready for release.
* **Test result analysis**: Or Defect Analysis, is done by the development team usually along with the client, in order to decide what defects should be assigned, fixed, rejected (i.e. found software working properly) or deferred to be dealt with later.
* **Defect Retesting**: Once a defect has been dealt with by the development team, it is retested by the testing team.
* **Regression testing**: It is common to have a small test program built of a subset of tests, for each integration of new, modified, or fixed software, in order to ensure that the latest delivery has not ruined anything, and that the software product as a whole is still working correctly.
* **Test Closure**: Once the test meets the exit criteria, the activities such as capturing the key outputs, lessons learned, results, logs, documents related to the project are archived and used as a reference for future projects.

**Types of Testing :**

Different testing types that are carried out in this project are :-

**Unit Testing**

The first test in the development process is the unit test. The source code is normally divided into modules, which in turn are divided into smaller units called units. These units have specific behavior. The test done on these units of code is called unit test.

Unit test depends upon the language on which the project is developed. Unit tests ensure that each unique path of the project performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**System Testing**

Several modules constitute a project. If the project is long-term project, several developers write the modules. Once all the modules are integrated, several errors may arise. The testing done at this stage is called system test.

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

Testing a specific hardware/software installation. This is typically performed on a COTS (commercial off the shelf) system or any other system comprised of disparent parts where custom configurations and/or unique installations are the norm.

**Other types of testing’s are :**

**Functional Testing**

Functional test can be defined as testing two or more modules together with the intent of finding defects, demonstrating that defects are not present, verifying that the module performs its intended functions as stated in the specification and establishing confidence that a program does what it is supposed to do.

**Acceptance Testing**

Testing the system with the intent of confirming readiness of the product and customer acceptance.

**Ad Hoc Testing**

Testing without a formal test plan or outside of a test plan. With some projects this type of testing is carried out as an adjunct to formal testing. If carried out by a skilled tester, it can often find problems that are not caught in regular testing. Sometimes, if testing occurs very late in the development cycle, this will be the only kind of testing that can be performed. Sometimes ad hoc testing is referred to as exploratory testing.

**Alpha Testing**

Testing after code is mostly complete or contains most of the functionality and prior to users being involved. Sometimes a select group of users are involved. More often this testing will be performed in-house or by an outside testing firm in close cooperation with the software engineering department.

**Beta Testing**

Testing after the product is code complete. Betas are often widely distributed or even distributed to the public at large in hopes that they will buy the final product when it is released.

**Automated Testing**

Software testing that utilizes a variety of tools to automate the testing process and when the importance of having a person manually testing is diminished. Automated testing still requires a skilled quality assurance professional with knowledge of the automation tool and the software being tested to set up the tests.

**Black Box Testing**

Testing software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as a specification or requirements document.

**Measurement in software testing :**

Usually, quality is constrained to such topics as [correctness](http://en.wikipedia.org/wiki/Correctness_(computer_science)), completeness, [security](http://en.wikipedia.org/wiki/Computer_security_audit), but can also include more technical requirements as described under the [ISO](http://en.wikipedia.org/wiki/International_Organization_for_Standardization) standard [ISO/IEC 9126](http://en.wikipedia.org/wiki/ISO/IEC_9126), such as capability, [reliability](http://en.wikipedia.org/wiki/Reliability_engineering), [efficiency](http://en.wikipedia.org/wiki/Algorithmic_efficiency), [portability](http://en.wikipedia.org/wiki/Porting), [maintainability](http://en.wikipedia.org/wiki/Maintainability), compatibility, and [usability](http://en.wikipedia.org/wiki/Usability).

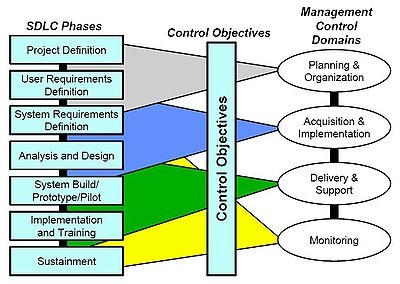
There are a number of frequently-used software measures, often called *metrics*, which are used to assist in determining the state of the software or the adequacy of the testing.

**MANAGEMENT AND CONTROL :**

The SDLC phases serve as a programmatic guide to project activity and provide a flexible but consistent way to conduct projects to a depth matching the scope of the project. Each of the SDLC phase objectives are described in this section with key deliverables, a description of recommended tasks, and a summary of related control objectives for effective management. It is critical for the project manager to establish and monitor control objectives during each SDLC phase while executing projects. Control objectives help to provide a clear statement of the desired result or purpose and should be used throughout the entire SDLC process. Control objectives can be grouped into major categories (domains), and relate to the SDLC phases as shown in the figure.

To manage and control any SDLC initiative, each project will be required to establish some degree of a [Work Breakdown Structure](http://en.wikipedia.org/wiki/Work_Breakdown_Structure) (WBS) to capture and schedule the work necessary to complete the project.

The following diagram describes three key areas that will be addressed in the WBS in a manner established by the project manager.



The upper section of the [work breakdown structure](http://en.wikipedia.org/wiki/Work_breakdown_structure) (WBS) should identify the major phases and milestones of the project in a summary fashion. In addition, the upper section should provide an overview of the full scope and timeline of the project and will be part of the initial project description effort leading to project approval.

The middle section of the WBS is based on the seven systems development life cycle (SDLC) phases as a guide for WBS task development. The WBS elements should consist of milestones and “tasks” as opposed to “activities” and have a definitive period (usually two weeks or more). Each task must have a measurable output (e.x. document, decision, or analysis). A WBS task may rely on one or more activities (e.g. [software engineering](http://en.wikipedia.org/wiki/Software_engineering), [systems engineering](http://en.wikipedia.org/wiki/Systems_engineering)) and may require close coordination with other tasks, either internal or external to the project.

**Baselines in the SDLC**

Baselines are an important part of the systems development life cycle (SDLC). These baselines are established after four of the five phases of the SDLC and are critical to the iterative nature of the model .

 Each baseline is considered as a milestone in the SDLC.

* functional baseline: established after the conceptual design phase.
* allocated baseline: established after the preliminary design phase.
* product baseline: established after the detail design and development phase.
* updated product baseline: established after the production construction phase.

Strengths and weaknesses of SDLC

Few people in the modern computing world would use a strict [waterfall model](http://en.wikipedia.org/wiki/Waterfall_model) for their systems development life cycle (SDLC) as many modern methodologies have superseded this thinking. Some will argue that the SDLC no longer applies to models like Agile computing, but it is still a term widely in use in technology circles. The SDLC practice has advantages in traditional models of software development, that lends itself more to a structured environment. The disadvantages to using the SDLC methodology is when there is need for iterative development or (i.e. web development or e-commerce) where stakeholders need to review on a regular basis the software being designed. Instead of viewing SDLC from a strength or weakness perspective, it is far more important to take the best practices from the SDLC model and apply it to whatever may be most appropriate for the software being designed.

A comparison of the strengths and weaknesses of SDLC:

|  |  |
| --- | --- |
| **Strength and Weaknesses of SDLC**[[10]](http://en.wikipedia.org/wiki/Systems_development_life-cycle#cite_note-Post.2C_G._2006-9) | |
| **Strengths** | **Weaknesses** |
| Control. | Increased development time. |
| Monitor large projects. | Increased development cost. |
| Detailed steps. | Systems must be defined up front. |
| Evaluate costs and completion targets. | Rigidity. |
| Documentation. | Hard to estimate costs, project overruns. |
| Well defined user input. | User input is sometimes limited. |
| Ease of maintenance. |  |
| Development and design standards. |  |

AN alternative to the SDLC is [rapid application development](http://en.wikipedia.org/wiki/Rapid_application_development), which combines prototyping, joint application development and implementation of CASE tools. The advantages of RAD are speed, reduced development cost, and active user involvement in the development process.

**OUTPUT & SNAKE GAME INTERFACE**

**Home Screen:-**

Snake game has a main screen displayed on its startup which we call **“Home Screen”**. Our Game Starts from here. Home Screen came along with various options which can be choosed by user by moving the cursor up and down using arrow keys and choose desired option by pressing Enter key.



Figure 1: Home Screen

**New Game:-**

This option is responsible for creating a new user profile and then starts the game from the beginning. If in case user select the name for his profile which currently exits i.e. used by another player then this screen displays appropriate message that “Currently profile exits with this name”.

**User can create New Profile with maximum 12 alphanumeric & special symbols.**

Given below 2 images showing what happen when user choose New Game option and try to create a new profile and another case in which user tries to create a profile with name which is currently used by another name.

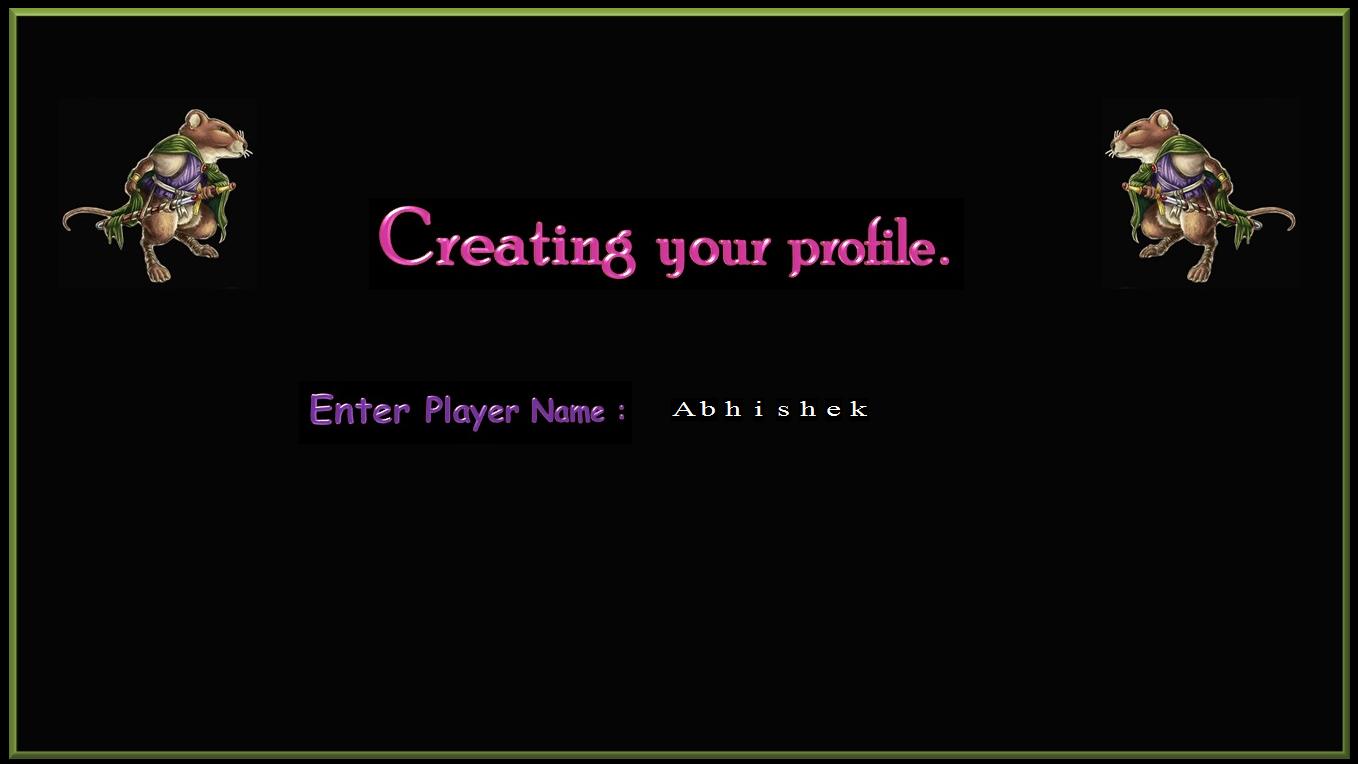


Figure 2: Creating new profile



Figure 3: Displaying error message when user tries to Create new profile which currently exist

**Load Game :-**

This option is part of “Auto Save Feature”. It loads the required user profiles which are saved by Auto save feature. User can load his\her valid profiles if they exist. In case if profiles do not exits then it show appropriate error message saying that “Invalid Profile Name”, otherwise if profile exits then it loads its data and then start the game.

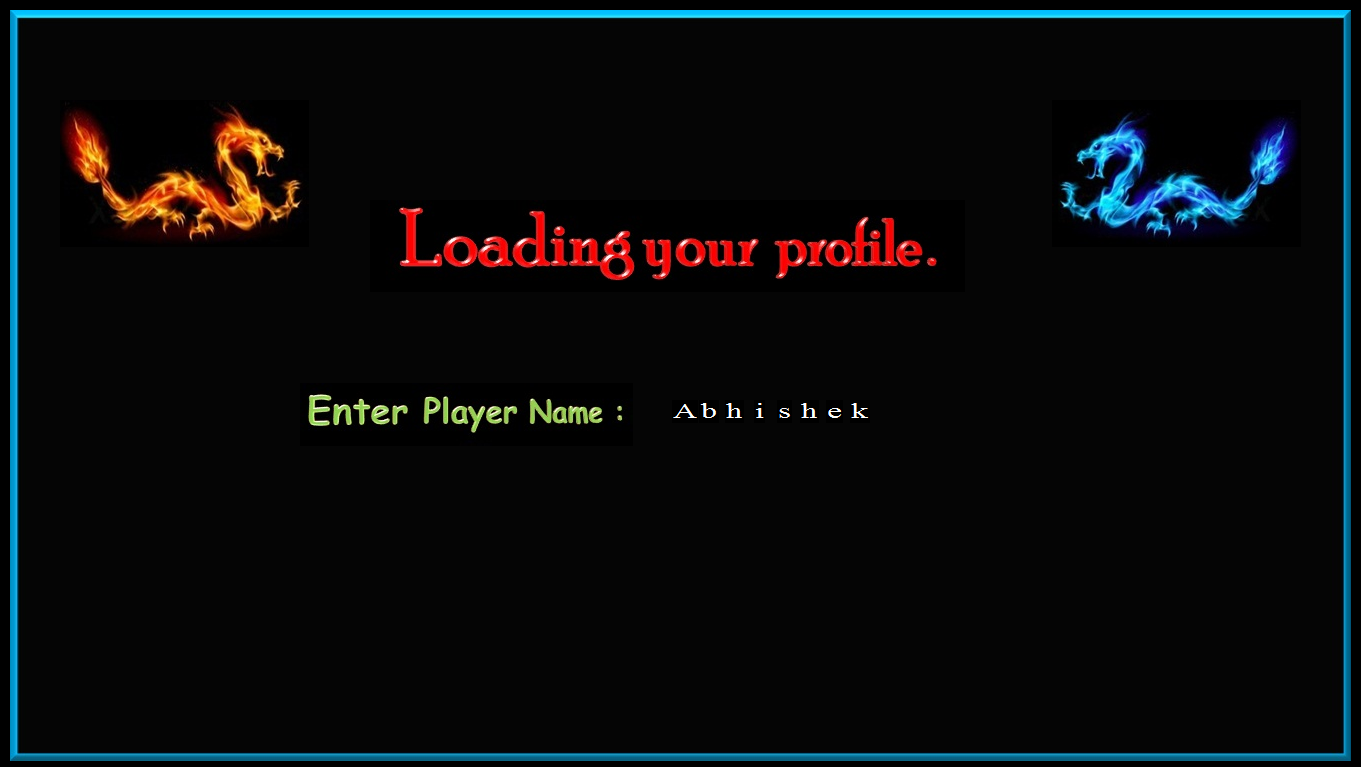


Figure 4: User loading profile named “Abhishek”



Figure 5: User got an error message when try to load profile which does not exist

**High Score :-**

This option is responsible for displaying the high score of **top 5 players** along with their Name, Rank, Life’s and level etc.



**Options :-**

Figure 6: High Score Menu

Options menu is responsible for changing speed setting of the snake. This menu has two options:

1. Speed settings
2. Back
3. **Speed Settings :**

This option shows the **“Speed Setting’s Menu”**, and contains three options for changing speed of the snake.

1. Beginner

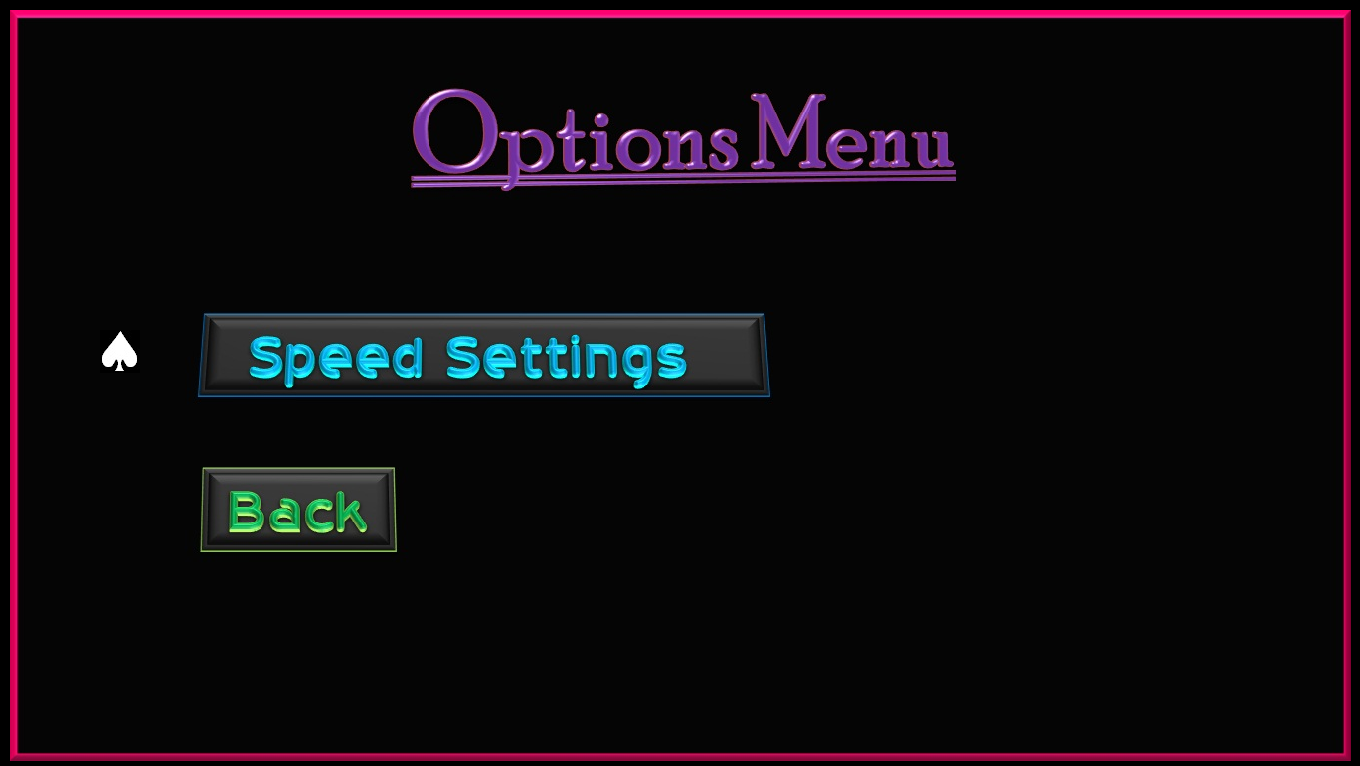
2. Intermediate

3. Advanced

User can choose any of these acc. to his convenience.

1. **Back :**

This option returns the user back to the **Home Screen**. User can also use **Esc** key for this operation.



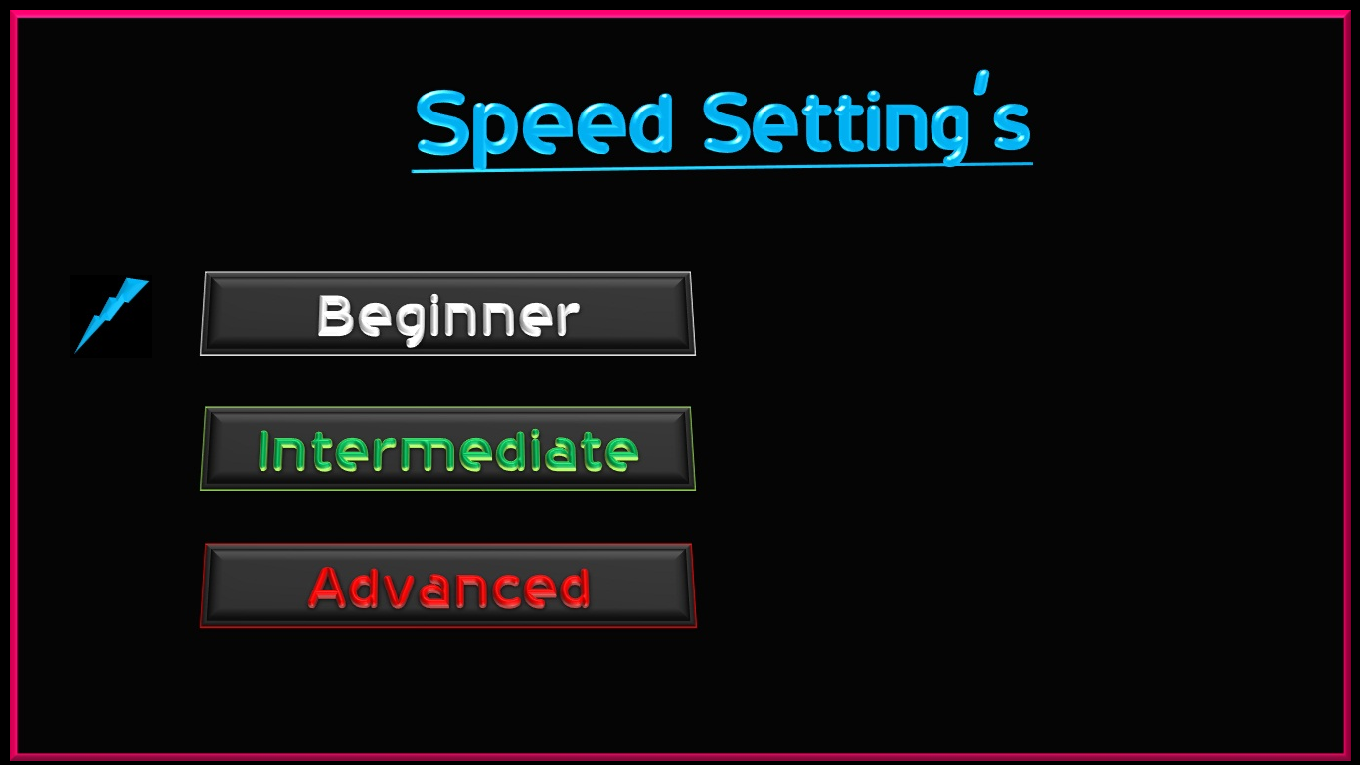


Figure 7: User choose options menu

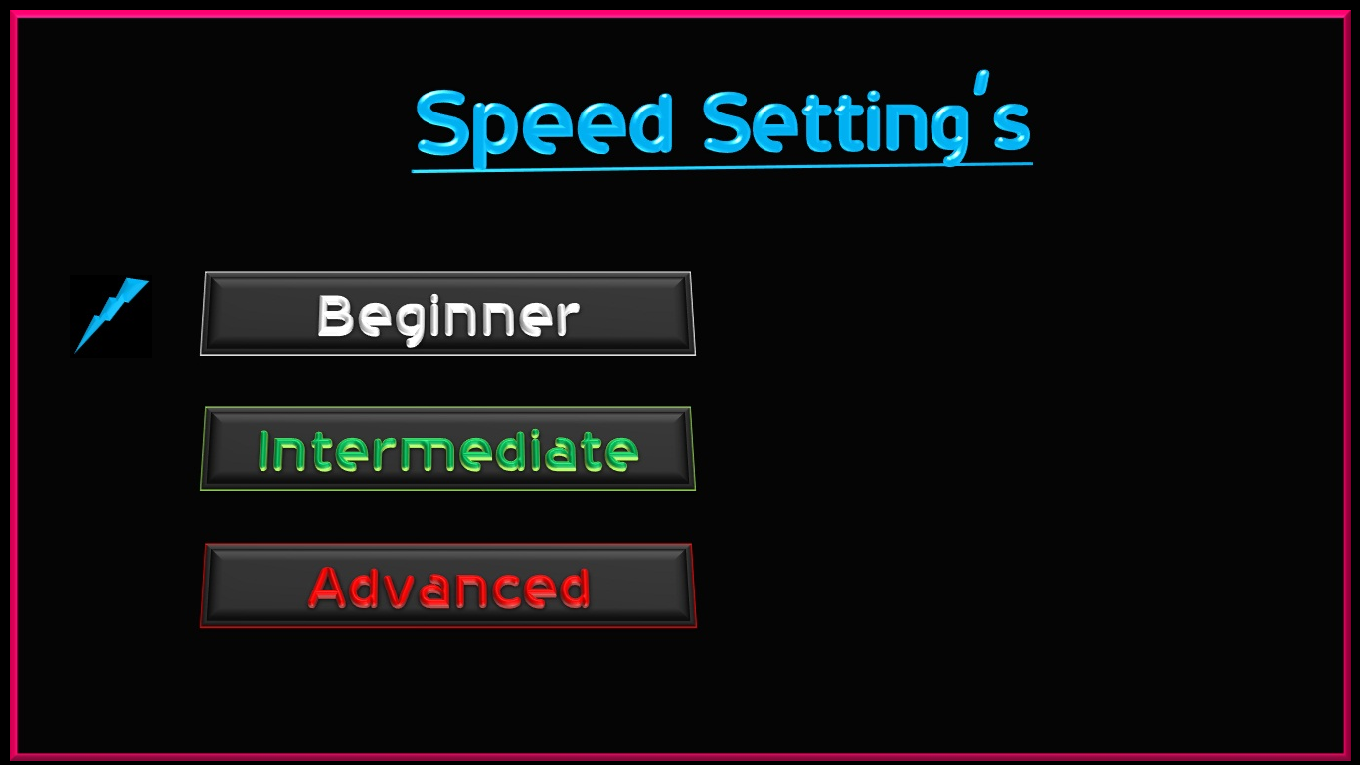


Figure 8: User choose Speed Setting’s option

**About Game :-**

This option is a part of “**Help Menu**”. It introduces the user with the characters and the controllers necessary to operate the snake.

It contains three options:

1. Controllers
2. Characters
3. Exit



Figure 9: User choose “About game” option

1. **Controllers:**

This options contains list of controllers required for handling snake and other basic operations in this game

1. **Characters:**

This option contains two sub options

1. Preys
2. Predators
3. **Preys:**

Preys are the birds/Animals which snake can hunt. Or in other words these are the snake food.

1. **Predators:**

These are those birds/Animals that **Hunts snake**.

1. **Exit:**

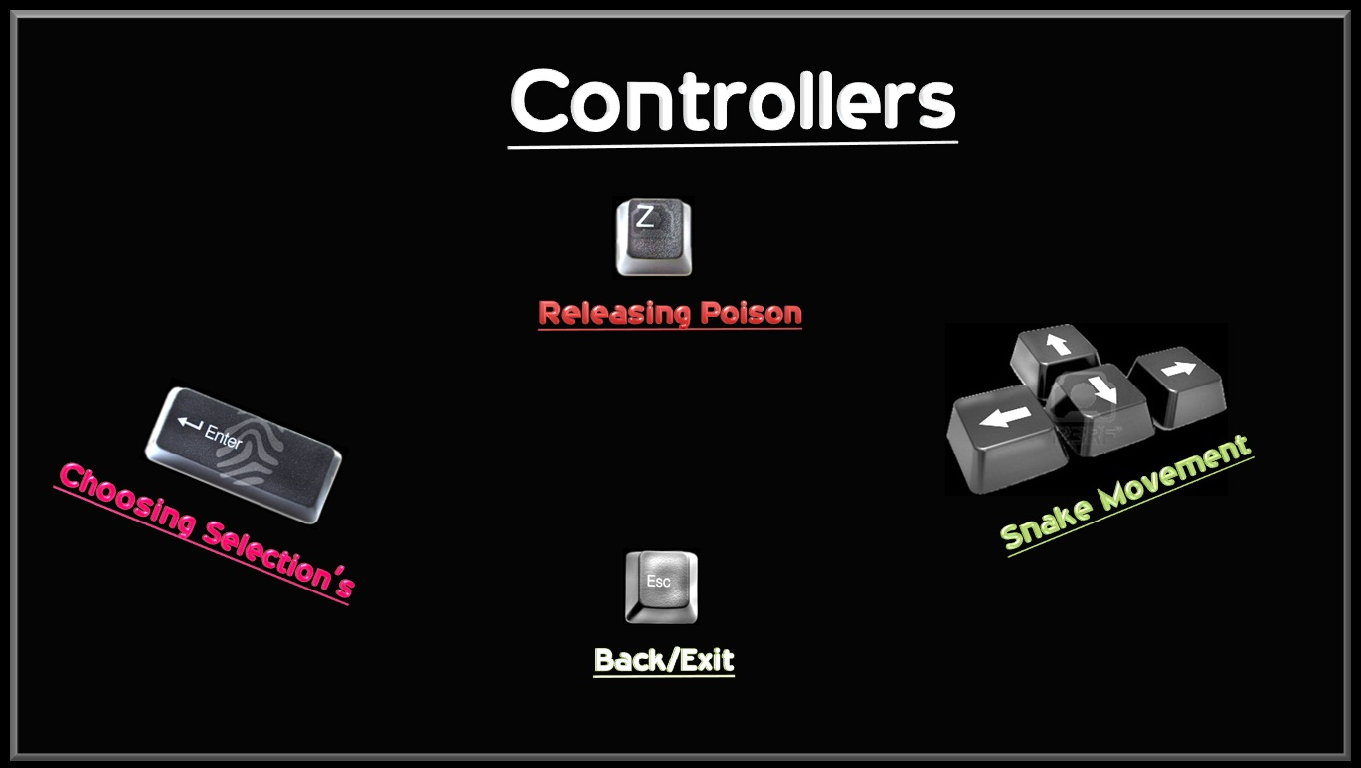
This option brings user back to “Home Screen”.

Figure 10: User choose “Controllers” option



Figure 11: User choose “Characters” option

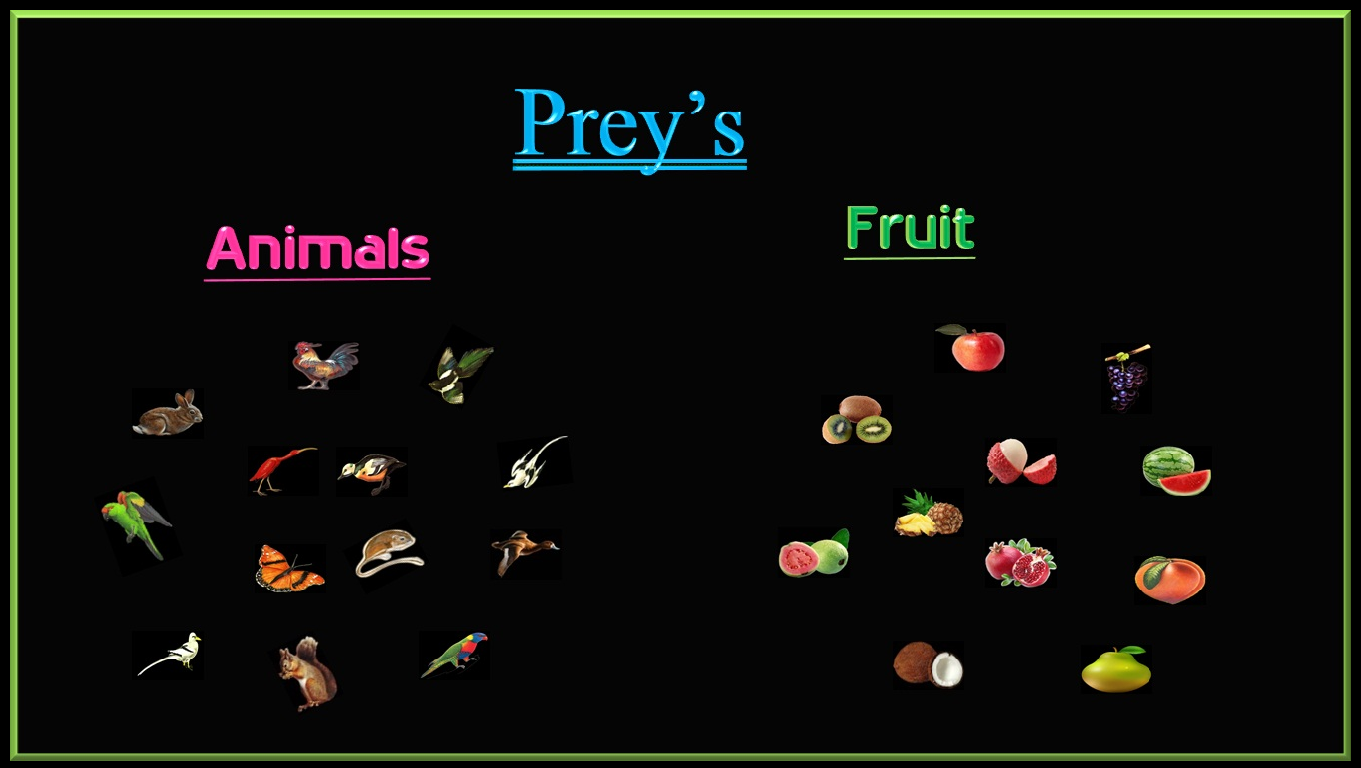


Figure 11.01: User choose “Preys” option

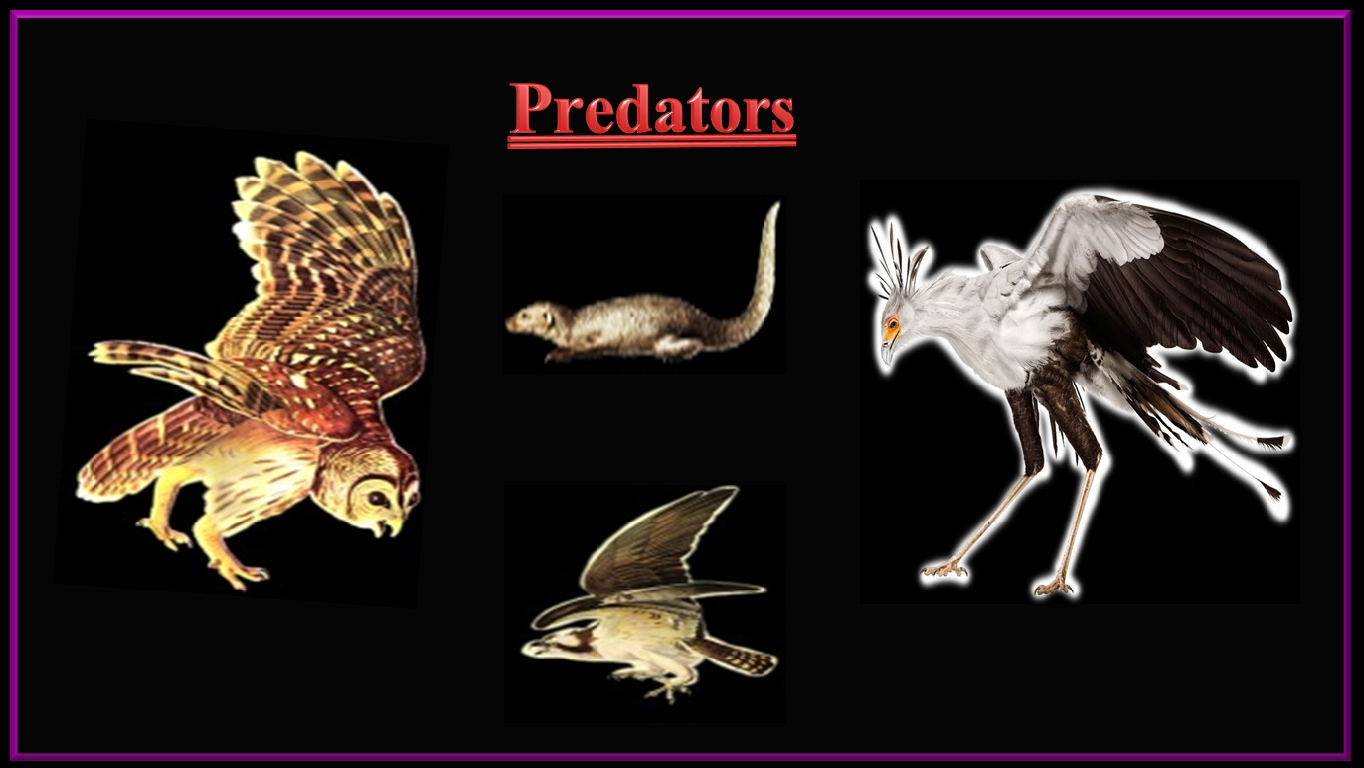


Figure 12: User choose “Predators” option

**Credits :-**

Shows Credits Menu.



Figure 13: User choose “Credits” option

**Exit :-**

This option Q**uits** the game.

Figure 14: Exiting Game

**Play Screen :-**

This is the screen where the game is played. This screen shows only when user choose **“New Game”** or **“Load Game** “option. We further divide this screen into two parts:

1. Header Screen
2. Playing Area
3. **Header Screen:**

Header screen is the big rectangle on the top of the screen containing various options like **Player, Level, Life’s, Power, High Score, Score, Food Eaten, Time Left etc.**

1. **Playing Area:**

Playing Area is the portion of screen surrounded by the dragons boundary, where snake roams with the aim of hunting his preys and prevent from the predators.



Figure 15: Play Screen

**CONCLUSION**

Snake game is an interactive 2D game application developed using c++ for MS-Windows Platform. We made this game by going through various phases of SDLC like analysis, design; coding and testing etc. the sole aim for developing was to understand and adopt System Analysis and Design Concepts and implement them in this project.

Now we have a game in which there is a snake which can be controlled by the user using arrow keys which roams on the screen to catch the preys. Preys are in the form of animals which moves along with various speeds and different paths. Snake had given **poison in the form of fire bomb** which is released by pressing the ‘**Z’ button** form the keyboard which helps the snake to catch the objects which are far away from the snake.

Our game has also included two main features which are **auto save** feature and the **Encryption** of profiles data to secure unauthorized modification by the users.

*We achieve most of our goal.*

**BIBLIOGRAPHY**

For making this game here are the some books and resources which help us a lot:

* Let us C by **Yashwant Kanetkar**
* Object Oriented programming with c++ by **E.Balaguruswamy**
* **Turbo C++** help manual
* Fundamental of computer and programming in C by **S.K. Jha**
* Object oriented programming in C++ by **Sumita Arora**
* **Google**