

Project Report



HERCULIAN CHESS



Project Briefing

- ▣ Project taken up by the team is to simulate a user-friendly environment to play a game of chess between two players.
- ▣ Ideally, the piece of software should ensure that the game is played according to the standard rules of the chess game and report the end-game scenarios.

Development Phase

- ▣ Project Timeline:

Start Date : 27th Sep 2011

End Date : 12th Nov 2011 (Last Date Modified)

- ▣ Modules of the Development Phase:

- Algorithm Module
- GUI Module
- Debugging Module

Module-wise Description

▣ Algorithms Module :

- This module is related to the basic algorithm with which the software was created or rather simply put : what happens behind the screen.
- This module consisted of the major decisions such as what classes to use and what are the data members to be allocated so that all the possible end-game scenarios can be reported and all special moves can be incorporated.
- Timeline :
 - ▣ Base Source Code : 27th Sep 2011 to 21st Oct 2011
 - ▣ Modifications : 04th Nov 2011 to 10th Nov 2011

Classes Description

Class Pos :

| Data Member | Type | Use |
|-------------|---------|---|
| x | Integer | Stores x position of the square |
| y | Integer | Stores y position of the square |
| colour | Integer | Stores the color of the square |
| coinId | Integer | Stores the coinId of the piece (if any) |
| coinIndex | Integer | Stores the coinIndex of the piece (if any) :: required mainly for post pawn promotion scenarios |

Classes Description

Class Movement:

| Data Member | Type | Use |
|------------------|---------|--|
| initPos | Pos | Stores initial position of a movement |
| finalPos | Pos | Stores final position of a movement |
| coinId | Integer | Stores coinId of moving coin |
| coinIndex | Integer | Stores coinIndex of moving coin |
| captureCoinId | Integer | Stores coinId of coin to be captured (if any) |
| captureCoinIndex | Integer | Stores coinIndex of coin to be captured (if any) |
| Colour | Intger | Stores colour of the coin to be moved |

Classes Description

Class Coin

| Data Member | Type | Use |
|-------------|--------------|--|
| curPos | Pos | Stores the current position of a piece |
| finalPos | | |
| coinRange | Array of Pos | Stores all the possible moves for a piece |
| coinId | Integer | Stores coinId of the piece |
| nRange | Integer | Stores total number of moves possible for each piece |
| nHistory | Integer | Stores the number of moves made by the piece in the game |
| mode | Integer | Stores if the piece is active |
| colour | Integer | Stores the color of the piece |

Important Global Variables

- ▣ The algorithm used is fool-proof in the sense that it maintains the record in different format in array of objects of the classes.
- ▣ `board[8][8]` is an array of `Pos` that stores the current game scenario of the chess board.
- ▣ `allCoins[32]` is an array of `Coin` that stores information about each piece on the board
- ▣ `currentMove` is an object of `Movement` that acts like the only tunnel thro which a move can be claimed

Implementation

- ▣ Whenever a move is claimed by a player, it is recognised by the GUI Module and it sends input to the Algorithms module
- ▣ `initializeMove()` is called which subsequently calls `makeMove()` which verifies the correctness of the movement claimed and returns appropriate value back to GUI implementing the corresponding changes in algorithms module
- ▣ GUI module makes the corresponding change on-screen and updates the range of all coins in algorithm module after checking for end-game scenarios.

Some Important Functions

- ▣ `updateRook()`
- ▣ `updateKnight()`
- ▣ `updateQueen()`
- ▣ `updateKing()`
- ▣ `updateBishop()`
- ▣ `updatePawn()`
- ▣ `verifyCheck()`
 - Note : As the name suggests they take care of updating the range of all the coins after every move is made

GUI Module

- ▣ GUI for the game is made in EzWindows with an extensive use of bitmaps
- ▣ GUI module maintains its own global and local variables and cannot access anything else from the Algorithms Module except the currentMove
- ▣ The two modules have been developed completely independent of each other
- ▣ Specific nomenclature of the bitmaps have been followed to ease the job.
- ▣ Timeline :
 - Start Date : 24th Oct 2011
 - End Date : 12th Nov 2011

Debugging Module

- ▣ As we had decided in the initiation of the project, debugging was allotted around 35-40% of the project time
- ▣ gdb in Ubuntu helped much in debugging segmentation faults
- ▣ One major confusion about the interchange of index numbers in board array was cleared
- ▣ Many logical errors were concerning uninitialized variables.
- ▣ Timeline :
 - Start Date : 04th Nov 2011
 - End Date : 11th Nov 2011

System Requirements ...

- ▣ Operating System:
UNIX Environment(Recommended)
(Tested in Ubuntu 11.04 and 11.10)
- ▣ Graphical User Interface:
Ez Windows
- ▣ Hardware Requirements:
 - 512 MB of RAM (Recommended)
 - Compatible with all Configurations of Graphic Card
 - Pointing Device(Required)

Future Improvements ...

- ▣ Undo previous move can be added as a special feature.
- ▣ The three fold repetition rule can be incorporated in the game.
- ▣ Load game facility can be provided to the user.
- ▣ The graphics can be improvised further to give it a 3D view.
- ▣ The current game is made to facilitate 2 player game. It can be improvised further to a single player game incorporating artificial intelligence.

Status of Completion ...

- ▣ The claimed project objectives were successfully completed.
- ▣ Final stage testing and debugging has been done.
- ▣ A trial run has been successfully done by neutral non-team members

Acknowledgement

- ▣ glchess in Ubuntu : .svg files of pieces were taken from the glchess game
- ▣ GIMP Image Editor : Used to edit and convert .svg to .xpm as is required by EzWindows
- ▣ Geany : The light IDE which was used for making the project.
- ▣ Friends: who helped us in realizing the minor and major flaws in the project

Consolidation Report...

Consolidated report (complete timeline)

| Name | Documentation | Discussions | Designing | Testing | Progr. | Misc. |
|-----------|---------------|-------------|-----------|---------|--------|----------|
| Guna | 06 Hrs | 13:15 Hrs | 25 Hrs | 30 Hrs | 22 Hrs | 20.5 Hrs |
| Sushant | 06 Hrs | 13:15 Hrs | 07 Hrs | 25 Hrs | 12 Hrs | 2.5 Hrs |
| Himanshu | 00 Hrs | 13:15 Hrs | 06:15 Hrs | 05 Hrs | 10 Hrs | 2 Hrs |
| Hardik | 00 Hrs | 13:15 Hrs | 05 Hrs | 10 Hrs | 10 Hrs | 2 Hrs |
| Kranthi | 00 Hrs | 12:45 Hrs | 01 Hr | 0 Hrs | 03 Hrs | 00 Hrs |
| Ritesh | 00 Hrs | 11:15 Hrs | 01 Hr | 0 Hrs | 04 Hrs | 00 Hrs |
| Indramoni | 00 Hrs | 09:45 Hrs | 03 Hrs | 0 Hrs | 0 Hrs | 00 Hrs |

Individual Contributions ...

- ▣ **Himanshu Roy:**
 - Base codes:
 - ▣ `updateRook();`
 - ▣ `insufficientMoves();`
 - ▣ `initialize();`
 - Modifications:
 - ▣ `updateKing();`
 - ▣ Debugging all the individual functions.
 - ▣ Image editing of various bitmaps and converting them to .xpm

Individual Contributions ...

▣ Hardik Kothari:

Base Codes:

- ▣ `updateKnight();`
- ▣ `verify_checkmate_stalemate();`
- ▣ `modifyBoard();`
- ▣ `undoBoard();`

■ Modifications:

- ▣ `updateQueen();`
- ▣ Debugging of all the individual functions written.
- ▣ Commenting of all the source codes.

Individual Contributions ...

- ▣ **Sushant Hiray:**
 - Base Codes:
 - ▣ `updatePawn();`
 - ▣ `updateFiftyMoveRule();`
 - ▣ `updateMakeMove();`
 - ▣ `updateBishop();`
 - Modifications:
 - ▣ `updateKing();`
 - ▣ Debugging of various functions. Almost all the `class_decln` errors were debugged.
 - ▣ Debugging of all the individual functions written.

Individual Contributions ...

- ▣ Ritesh Kakade:
 - Base Codes:
 - ▣ `updateKing();`

Individual Contributions ...

- ▣ **Kranthi Kumar:**

- ▣ Base Codes:

- ▣ `updateQueen();`

Individual Contributions ...

- ▣ **Indramoni Rout:**
 - Base Codes:
 - ▣ He tried writing the `updateBishop()` function but couldn't complete it fully.
 - ▣ He attended many Team Meetings.

Individual Contributions ...

- ▣ **Guna Prasaad:**
 - Base Codes:
 - ▣ verifyCheck();
 - ▣ Class_decln.cpp;
 - ▣ Complete GUI
 - Modificatons:
 - ▣ Class_decln.cpp;
 - ▣ All range update functions.
 - ▣ Debugging of all individual functions , GUI , class_decln
 - .