PROJECT DOCUMENTATION

# Project: BOWMAN

TEAM: 1. Anirudh Vemula -110050055

2. Vamsidhar Y -110050051

3. Rama Krishna .B -110050054

# Description of the problem:

Aim of the project is to design an archery game using Scheme .The reason of picking up this project is not only for fun but also for the learning that comes along with it.Our project creates a turn based archery game which can be played by two human players or with a human and a computer player (AI player).The game consists of two archers separated by a distance and shoot against each other in turns. The game ends when an archer kills the other. This project gives us an understanding of how to apply basic laws of Physics (like, Projectile motion in our case) through Scheme.

# Modules and Graphics packages used:

1. 2htdp/universe
2. 2htdp/image
3. Racket/gui
4. Sounds package in racket/gui

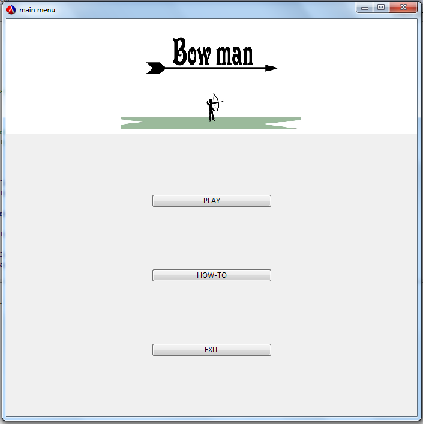
# Overall design of the program:

The introductory menu and the subsequent windows are created using **racket/gui package**. The game interface is created using **2htdp/universe** module

The following objects are used to create the windows

* frame%
* canvas%
* button%
* bitmap%

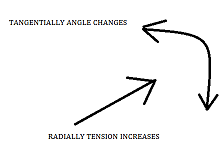
Windows are created using the frame% object on which the buttons are placed which call upon subsequent windows a canvas is also used to depict a bitmap image in all the windows.



When the single/double player button is pressed the main program calls upon a function, which is included in another .rkt file. The function called, opens a new window which is the world this is the game interface in which all the inputs are recorded and outputs are shown using 2htdp/universe module.

All the inputs in the game are through mouse. The mouse events corresponding are

Click and drag, button up and button down.



WHEN YOU CLICK AND DRAG THE MOUSE RADIALLY, THE TENSION CHANGES AND TANGENTIALLY, THE ANGLE CHANGES.BOTH OF THESE INPUTS ARE TAKEN SIMULTANEOUSLY.

The inputs taken correspond to two different attributes namely angle and tension.

Angle decides the orientation of bow with the horizontal axis and the tension depicts the tension in the bow string .These inputs are taken through the mouse event “drag” .When the user releases the mouse button (button up event) the current mouse pointer coordinates are used to calculate the angle and the tension. All these mouse events are executed through a function on-mouse.

The subsequent motion of the arrow is shown using the on-tick function which redraws a given scene every 1/28th of a second .Basic laws of projectile motion are included in this function to make sure that the arrow follows a parabolic path. The on-tick function also records where the arrow has hit in separate variables corresponding to hitting the ground or the opponent. Some flags are included to differentiate between the shots of one archer from the other.

The same functions as above execute the shot made by the other archer. Another function to-draw gives output as images /scenes (using **2htdp/image module**) that are to be drawn at each tick (28th part of a second) subject to the conditions prevailing.

The crux of the interface is executed by the legendary big bang function of the **2htdp /universe** module which takes the on-mouse, on-tick and to-draw functions as its arguments and ensures that the inputs are taken through on-mouse function and the output (scene) of the to-draw function is drawn through the on-tick function at every tick.

Single player mode differs from the double player mode as in the second archer being controlled by the computer through an algorithm, which is self-rectifying. The programs for the single player and double player are written in two different files (.rkt), which are included in the main file.



# Algorithm for AI player:

The position of the arrow when it hits the ground is recorded through the on-tick function. This value is used by the AI player to decide the inputs for its next shot. For ex, when the arrow goes beyond the opponent, the Ai player changes his inputs, i.e. angle and tension accordingly so as to decrease the range of his arrow. The same is done in the case when the arrow falls short of his opponent. The wind in the air supplies a random element to the accuracy of the AI player as without the wind his accuracy increases with turns. In short, the AI player has a memory of 1 turn and uses it to increase his accuracy

# Limitations and bugs:

1. The wind in the air which acts as a random element can prove to be too random sometimes (rarely) making the AI player inaccurate.
2. The image of the hands and the torso of the archer look a bit clumsy when the bow is stretched too long.

# Other Points of Interests:

* The game exists in a very interactive manner as in the changes in the inputs can be seen instantaneously on the window. For example, when we change the angle and tension the changes made, are visible on the archer’s and the bow’s orientation continuously.
* Both the inputs i.e. angle and tension is taken in a single mouse-event (“drag”) reducing the time taken for the archer to set his inputs. This makes the game more interesting and less boring.
* The state of the game at any point of time is described in the program using two flags minimizing the need for multiple variables to serve the purpose. For example, actions of one archer change the flags from 0 to 1 and the other archer’s actions change the flags from 1 to 0 thus repeating the process in the form of a loop until the game ends.
* Three graphic packages namely, **racket/gui, 2htdp/universe and 2htdp/image** are integrated to form the complete game. Bitmaps are included in the game at some instances to make it more presentable.
* Sounds are included in the game to make the game more realistic and more entertaining.