15-466 Computer Game Programming
Fall 2011
Project 4

For this assignment you will be implementing a checkers game from the ground up for a mobile device. This project will involve several components: visualization of the game, user interface, the game mechanics (rules), and an AI player. We have simplified the rules of the game slightly so that you don't have to focus on the AI as much as the other parts.

Rules

I've taken the rules of the game of checkers from here (credit to Erik Arneson):
http://boardgames.about.com/cs/checkersdraughts/ht/play_checkers.htm
I've put our changes in **bold**. We are basically disallowing multiple jumps.

1. Checkers is played by two players. Each player begins the game with 12 colored discs. (Typically, one set of pieces is black and the other red.)
2. The board consists of 64 squares, alternating between 32 dark and 32 light squares. It is positioned so that each player has a light square on the right side corner closest to him or her.
3. Each player places his or her pieces on the 12 dark squares closest to him or her.
4. Black moves first. Players then alternate moves.
5. Moves are allowed only on the dark squares, so pieces always move diagonally. Normal pieces are always limited to forward moves (toward the opponent).
6. A piece making a non-capturing move (not involving a jump) may move only one square.
7. A piece making a capturing move (a jump) leaps over one of the opponent's pieces, landing in a straight diagonal line on the other side. Only one piece may be captured in a single jump. **We are not allowing multiple jumps.**
8. When a piece is captured, it is removed from the board.
9. If a player is able to make a capture, there is no option -- the jump must be made. If more than one capture is available, the player is free to choose whichever he or she prefers.
10. When a piece reaches the furthest row from the player who controls that piece, it is crowned and becomes a king. **You should some how show the piece is "crowned".**
11. Kings are limited to moving diagonally, but may move both forward and backward. (Remember that single pieces, i.e. non-kings, are always limited to forward moves.)
12. In addition to moving, kings may also capture forward and backward (diagonally), **but still only one jump is allowed**.
13. A player wins the game when the opponent cannot make a move. In most cases, this is because all of the opponent's pieces have been captured, but it could also be because all of his pieces are blocked in.

Visualizations
You are free to make the appearance of the game whatever you like as long as it is still relatively clear how to play checkers on it. It is possible to make all of the visuals from within Unity by using basic shapes (cubes, cylinders, etc.) and coloring them with basic rgb materials. You are free to make your own textures for your objects outside of Unity (Photoshop, Gimp, etc). You may also model your own objects (Maya, Blender, etc.) or use content from online (as long as you cite them). Here are some links to 3D models should you choose to use them.
http://www.blendswap.com/
http://www.turbosquid.com/
http://e2-productions.com/repository/modules/PDdownloads/topten.php?list=hit

Interface
The basic interface of the game will involve tapping on the screen of a mobile device to select the piece you want to move and then tapping again to tell it where to go (in accordance with the rules). You may change the interface to work some other way, but if you do **your game must provide clear instructions on how the interface works**.
Conveniently, a single block of unity code allows you to do this on a tablet as well as with the mouse on your computer. Therefore, you should be able to test your code on your computer using the mouse and when you put it on a mobile device, the interface should work with no changes to the code. Here is the code for getting what GameObject the player has clicked on (this is the only code we will be giving you for the assignment).

```
//check if the user clicked
if(Input.GetMouseButtonDown(0)){
        //take the pixel clicked in the screen image and covert it to a ray
        Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
        RaycastHit hitInfo = new RaycastHit();
        //ray trace and return true if it collides with a GameObject's collider
        if(Physics.Raycast(ray, out hitInfo)){
                //when Physics.Raycast succeeds, it fills out the hitInfo object
                //get the GameObject that the collider belongs to
                GameObject g = hitInfo.collider.gameObject;
                //print out the GameObject's name
                Debug.Log(g.name);
        }
}
```

The way this code works is by raycasting out of the game's main camera through the pixel the user clicked on. The ray stops at the first collider it hits. If you want an object to not be clickable (the ray to pass through it) then turn off it's collider. On the other hand if you want the user to be able to click on it (like a game piece or board square) **make sure it has a collider component that covers all parts of the object that the user can click on**. This is generally set up by default when using a basic Unity object like cube or cylinder. However, it may not be the case for an arbitrary 3D model you download.

AI Player
You will have an AI player, which the user can play against. Your player must abide by the rules of the game and should play reasonably well. You may use any approach you would like for the AI though we recommend a game tree of some kind (minimax, negamax, alpha-beta pruning).

Extra Credit
We are hoping you will spend a substantial amount of time on the extra credit of this assignment. Extra credit will be awarded to students for adding audio or fancier visual effects to the game. You can also get extra credit for making use of mobile device hardware (accelerometer, gyro, cameras, etc) or coming up with a better or more fun user interface. We will be giving out up to 30% extra credit and the number points you get will be based on how cool your bells and whistles are compared to your classmates. Imagine you are writing a game for a competitive market, and your goal is to make the game as cool as possible, so that users choose your game over the others. Be creative and have fun!

Mobile Device
We have purchased Acer Iconia Tabs, which you will be able to use to try your game on a mobile device. These tablets have a 7-inch display and run android (honeycomb). The free version of Unity does not allow you to compile your code for mobile devices. We will have laptops in our lab (NSH 1612) which have the full version that you can use to do this. If you want to be able to do it yourself you can either purchase Unity or use your free 30-day trial. Also, if you would prefer to use your own android device (instead of ours) you may.
You should be able to make and debug your game using your own computers and then it should port easily to the mobile device. The only change I made was zooming out the main camera a bit to accommodate the smaller screen. The only exception would be if you choose to use some additional tablet hardware for the extra credit (then you may need to spend more time with the device).

What to submit

- A short write-up. I'm expecting less than a page and certainly no more than two (your game should speak for itself!). Briefly indicate how your AI works and then talk about anything interesting you implemented!
- A video showing your game in action. This can be recorded on your computer or on a mobile device.
- Your code

Grading

25%	The game state is visualized in an understandable way
20%	The user interface works and pieces move according to the rules of the game
35%	The AI player performs reasonably well
10%	The game runs on one of our mobile devices
5%	Your write-up
5%	Your video

Extra Credit will be given out as described above (up to 30%).