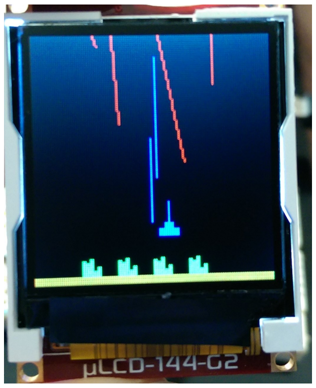
This project creates an Mbed version of Atari's classic Missile Command video arcade game that was wildly popular in the 1980's. In our Missile Command game, a city is being bombarded by missiles falling from the sky. A hovering aircraft (the player) flies just above the city and is able to shoot anti-missiles vertically to intercept the missiles threatening the city. If a missile gets past this guardian aircraft, it destroys a part of the city landscape. If all sections of the city are destroyed or if the aircraft itself is hit by a missile, it's Game Over! The objective of the game is for the player to fly the aircraft left and right across the city's landscape, intercepting as many missiles as possible for as long as possible. The more missiles are intercepted, the higher the score, and the player advances to more levels of the game. At each new level, the games gets more difficult: the missiles are generated at a higher rate and they fall at a faster speed.



In this project, you will be given shell code that implements parts of the game and you will complete the rest. The portion given to you is the missile generator, the city landscape display/undisplay maintainer, and aircraft display/undisplay features. You must complete the basic game functionality (described below) for 75 points, including 25 points (P2-1) for completing a doubly linked list module. You can earn additional points by successfully implementing extra features to enhance the basic game (as described below). These are worth 5 points per extra feature, for a maximum of 50 additional points (10 additional features). So, to get a 100% on this project, your mbed program must support the basic game and 5 additional features. Supporting up to 5 more features will earn you up to 25 points extra credit.

P2-1 (25 points): Doubly linked list implementation

The files `doubly_linked_list.h` and `doubly_linked_list.cpp` provide a shell for a utility module that will be used by the missile command game. The functions in this module create and manipulate doubly linked lists of nodes (e.g., inserting/deleting nodes, accessing nodes, creating/destroying the list). The file `doubly_linked_list.h` documents what each of the functions are supposed to do. Some of these functions are implemented in `doubly_linked_list.cpp`; you need to complete those that are not. (Note that all the code for this project should be written in C, even though the file extension is "cpp".)

A test bench has been provided to allow you to create traces of doubly linked list function calls for testing. The testbench is described in the file `LL_testbench_documentation.pdf`. A sample trace file is provided (`test1.txt`). You should create and run many different tracefiles to be sure that your code is correct.

This part of the project is due **Wednesday, 9 November 2016** at 5p.m.

P2-2 (50 points): Basic Game Functionality

These features must be implemented and working correctly to earn the baseline 50 points:

Aircraft movement: This uses the accelerometer to sense when the player tilts the circuit board in a given direction. Your program should use the data from the accelerometer to control the direction in which the aircraft should move. Your program should keep track of where the aircraft is on the screen and it should never go off the screen. (The uLCD is 128x128 pixels.)

Firing at missiles: When the player presses pushbutton PB2 (connected to pin P22), the aircraft should shoot an anti-missile vertically. Your program should display the vertical trajectory of each anti-missile as a blue line (as shown in the demo version) over time, eventually stopping at the top of the screen or when it intercepts a missile, at which point your program should make the blue line disappear. Your program needs to keep track of the position of the aircraft and of each anti-missile that is currently in flight (i.e., before it hits a missile or the top of the screen).

Detecting missile intercept and animating explosion: Your program needs to determine whether any of the anti-missiles that are currently traveling upward are within a certain radius of any incoming missile. (Initially, the radius should be 10 pixels, but make it adjustable so that you can set it larger or smaller depending on the current level of difficulty of the game.) If so, it should draw an explosion as a simple circle to indicate the collision and then notify the missile generator that the missile has exploded (by marking its status as `MISSILE_EXPLODED`) which will cue the missile's deletion and erasure from the screen on the next `missile_generator` call.

Detecting and animating city destruction: Your program should determine when a missile has hit a city segment (hint: `city_get_info` may be useful). If so, it should draw an explosion as a simple circle to indicate destruction and notify the landscape maintainer to destroy and undisplay that city segment using `city_destroy`. The circle showing the city segment explosion should be a different color than the one used to show missile explosions.

Detecting Game Over: After all four city segments have been destroyed or if the aircraft is hit by a missile, the game ends with a "Game Over" message displayed on the screen.

Displaying score: display the number of missiles intercepted so far on the uLCD screen in the top right corner.

Advancing levels: After ten missiles have been intercepted while the game has been played at a certain level, it should advance to the next level. At each level, your program should set the rate at which the missiles are generated and the missile speed, increasing each at each advancement. It should also set the intercept radius to be smaller at each advancement. The current level number should be displayed on the uLCD in the top left corner.

Force level advance: When pushbuttons PB1 and PB3 are pressed simultaneously, the game should advance to the next level. (This will make it easier to test and grade the "Advancing levels" feature.)

Documentation: be sure to document your code. Points will be deducted for poorly documented code.

This part of the project is due **Monday, 21 November 2016** at 5p.m.

P2-2 (up to 50 points): Extra Features

You can implement a subset of these features to earn an additional 5 points per feature, up to a maximum of 50 points:

More complex explosion animation – when a missile hits the city or is intercepted, more complex graphics can be used to animate the explosion (e.g., a series of concentric circles radiating out of the impact point).

Multiple lives – allow a player to earn multiple lives so that the game continues with a regeneration of the entire city landscape after the player survives for a certain amount of time.

Change aircraft shape to something fancier.

Add MIRVs – these are multiple independently targetable reentry vehicles; a single missile splits into a small number of missiles that spray out in multiple directions. The altitude of MIRV deployment could decrease at higher levels of game play.

Add new hardware to do something interesting such as using the shiftbrite LED to indicate some status information with color.

Enable diagonal anti-missile trajectories: use a pushbutton to toggle between two modes: one where the aircraft moves left or right (the baseline mode) and one where the aircraft *rotates* left or right when the board is tilted. When the aircraft has rotated and pushbutton PB2 is pressed and fired, the trajectory will be diagonal, rather than vertical. (Be sure to limit the rotation to the 180 degree arc above the aircraft so that the city does not come under friendly fire).

Smart/steerable anti-missile: allow player to steer the direction of the anti-missile while it's in flight using the accelerometer.

Include a Game Menu for configuring the game (e.g. starting it at some level: Easy/Medium/Hard).

Keep track of game history and show in interesting way (e.g., highest score so far) – this requires that you reset the game using a pushbutton without using the mbed's reset button which reinitializes the entire program. When game ends, generate an interesting animation/video (e.g., a triumphant one when you get a new high score).

Add sound effects – be creative. These could accompany explosions, firing, advancing to a new level, gaining a new life, hitting a new high score, etc.
There is documentation on "Playing a Song with PWM using Timer Interrupts" here: http://developer.mbed.org/users/4180_1/notebook/using-a-speaker-for-audio-output/
Hello world song player code URL is http://developer.mbed.org/users/4180_1/code/song_demo_PWM/
Video of a media player: http://developer.mbed.org/users/4180_1/notebook/mpod---an-mbed-media-player/

Use a pushbutton to create a new feature that can be used a limited number of times: for example, a super anti-missile with a larger radius of intercept or that clears the screen, or speed boosters for anti-missiles or the aircraft.

To get more ideas, Google “Missile Command” to find playable versions of the original classic game from Atari. If the feature you would like to implement is not mentioned here, that's great, but be sure to check with an instructor or TA to see if it would count.

This part of the project is also due **Monday, 21 November 2016** at 5p.m.

Shell Code and API Support

The project shell code is available at https://developer.mbed.org/teams/ECE2035-TA/code/project_prototype_released_version/ Click the “Import this program” button (upper right) to bring this project into your own account.

Documentation for this shell code is available on T-square: **missile_command_project_api.pdf**.

Also, a feature checklist file named **P2-checklist.txt** is available on T-square. Put an X before each feature in the checklist that your program implements and that you will demonstrate to the TA.

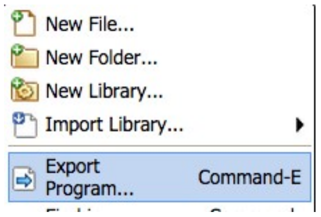
Project Submission

In order for your solution to be properly received and graded, there are a few requirements.

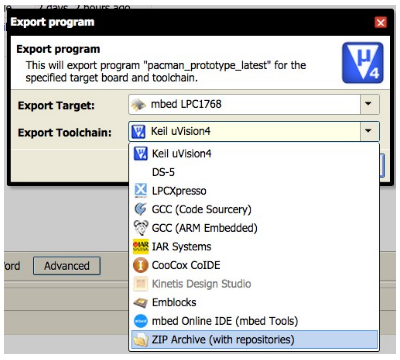
<u>For P2-1</u>: Upload `doubly_linked_list.h` and `doubly_linked_list.cpp` to T-square before the scheduled due date, **5:00pm on Wednesday, 9 November 2016** (the grace period is documented on T-square).

<u>For P2-2</u>:

1. Create a .zip archive of your project using the following steps:
2. In the Program Workspace (the left hand vertical scrolling window in the mbed compiler), right-click the project name of your project and select "Export Program..."



3. A box will pop up. Choose "ZIP Archive" in the Export Toolchain menu:



Export program

Export program

This will export program "pacman_prototype_latest" for the specified target board and toolchain.

Export Target: mbed LPC1768

Export Toolchain: Keil uVision4

4. Select “Export” and another dialog box will pop up that will allow you to save the file. Name this archive **`P2.zip`**.

5. Upload **`P2.zip`** and **`P2-checklist.txt`** to T-square before the scheduled due date, **5:00pm on Monday, 21 November 2016** (the grace period is documented on T-square).

6. After your project zip file is uploaded to T-square, meet with a TA and demo your game on **your** hardware. While the TA is watching, you will download the files you submitted from T-square, compile them in the mbed cloud compiler, download the executable to your mbed, and then demonstrate all of the features of your game. Bring a printout of the checklist you submitted showing which features you successfully implemented so that the TA can confirm them. This must be completed by **Friday, 2 December 2016.**

You should design, implement, and test your own code. There are many, many ways to code this project, and many different possibilities for timing, difficulty, responsiveness and general feel of the game. Your project should represent your interpretation of how the game should feel and play. Any submitted project containing code (other than the provided framework code) not fully created and debugged by the student constitutes academic misconduct.

Project Grading: The project weighting will be determined as follows:

part	*description*	*due date*	*percent*
	Missile Command Program		
P2-1	Doubly Linked List	Wed, 9 Nov 2016	25
P2-2	Baseline features	Monday, 21 November 2016	50
P2-2	Advanced Features		50
	total		125/100 maximum

<u>**Extra Extra-Credit:**</u> Here's another chance to earn extra credit (and the admiration of your fellow students). Create a short (< 1 minute) video clip highlighting the coolest feature(s) of your mbed project. If you enter a high quality video, you will earn an extra credit point on your overall course grade. (This is particularly helpful if you are on a letter grade boundary).

Your video clip must clearly identify the features that you are highlighting. To submit your entry, save your video clip as a **`.mov`** or **`.mp4`** file named "**`P2clip.mov`**" or "**`P2clip.mp4`**" and upload it by **Tuesday 29 Nov 2016** to T-square under Assignments > "Missile Command Highlights". We'll show a highlight reel of the video clips in lecture on the last day of classes with special recognition going to the top entries.

References

1. Mbed Pinout:
   http://mbed.org/nxp/lpc2368/quick-reference/

2. Mbed-NXP Pinout:
   http://mbed.org/users/synvox/notebook/lpc1768-pinout-with-labelled-mbed-pins/

3. NXP-LPC1768 User's Manual:
   http://www.nxp.com/documents/user_manual/UM10360.pdf