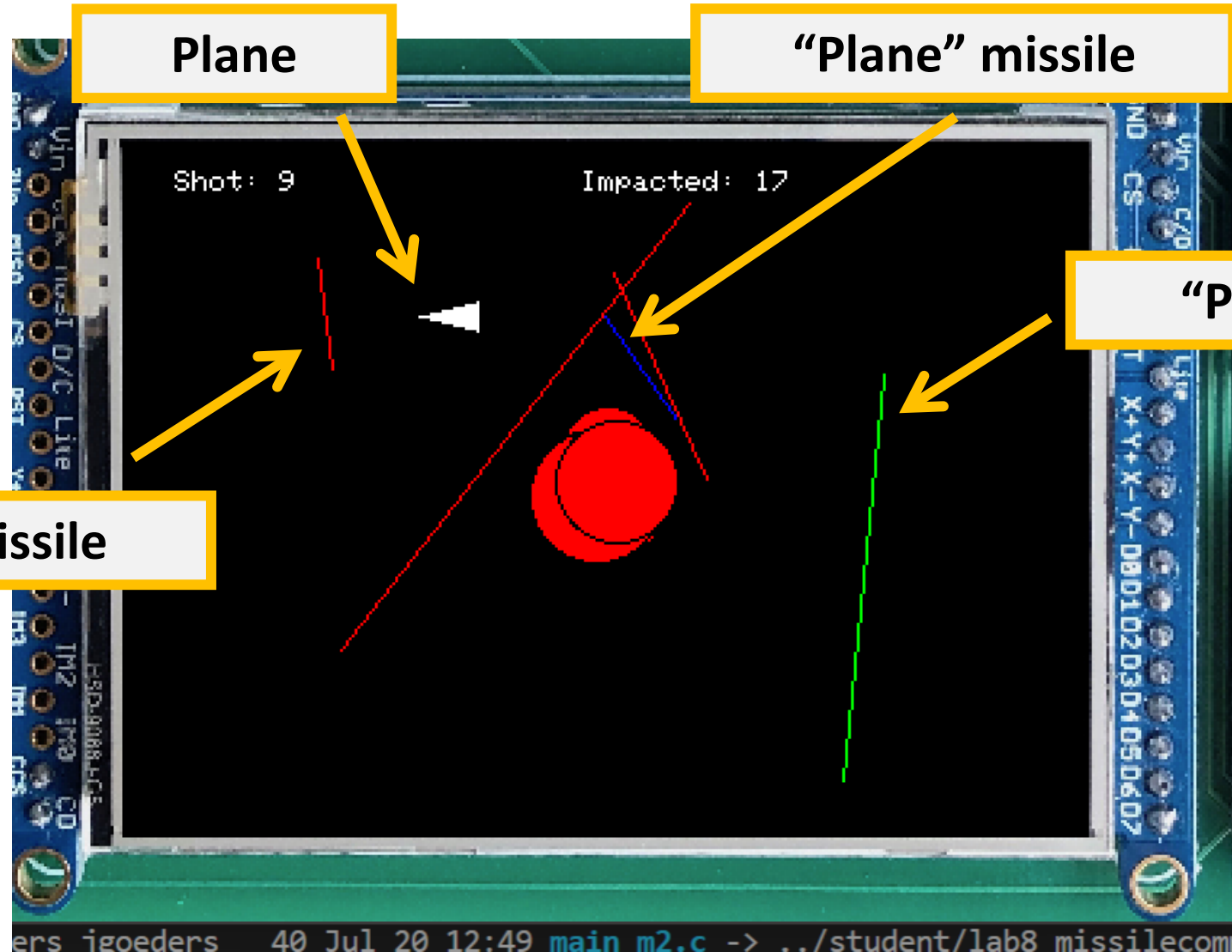


# Missile Command

ECEN 330

**BYU** Electrical & Computer  
Engineering

IRA A. FULTON COLLEGE OF ENGINEERING



# Milestone 1: Missiles

Every missile is its own state machine!

- Keep track of the missile current state and other properties in a struct

Tick function will accept a pointer to a missile struct:

```
void missile_tick(missile_t *missile);
```

Multiple different init functions depending on missile type:

- missile\_init\_enemy(missile\_t \*missile,..)
- missile\_init\_player(missile\_t \*missile,..)
- missile\_init\_plane(missile\_t \*missile,..)
- missile\_init\_dead(missile\_t \*missile,..)

```
/* This struct contains all information about a missile */
typedef struct {

    // Missile type (player, enemy, enemy plane)
    missile_type_t type;

    // Current state (the 'enum' will be defined in your missile.c file, so it's
    // just declared as an integer type here)
    int32_t currentState;

    // Starting x,y of missile
    uint16_t x_origin;
    uint16_t y_origin;

    // Ending x,y of missile, and the total length from origin to destination.
    uint16_t x_dest;
    uint16_t y_dest;
    uint16_t total_length;

    // Used to track the current x,y of missile
    int16_t x_current;
    int16_t y_current;

    // While flying, this tracks the current length of the flight path
    uint16_t length;

    // While flying, this flag is used to indicate the missile should be detonated
    bool explode_me;

    // While exploding, this tracks the current radius
    double radius;

    // Used for game statistics, this tracks whether the missile impacted the
    // ground.
    bool impacted;

} missile_t;
```



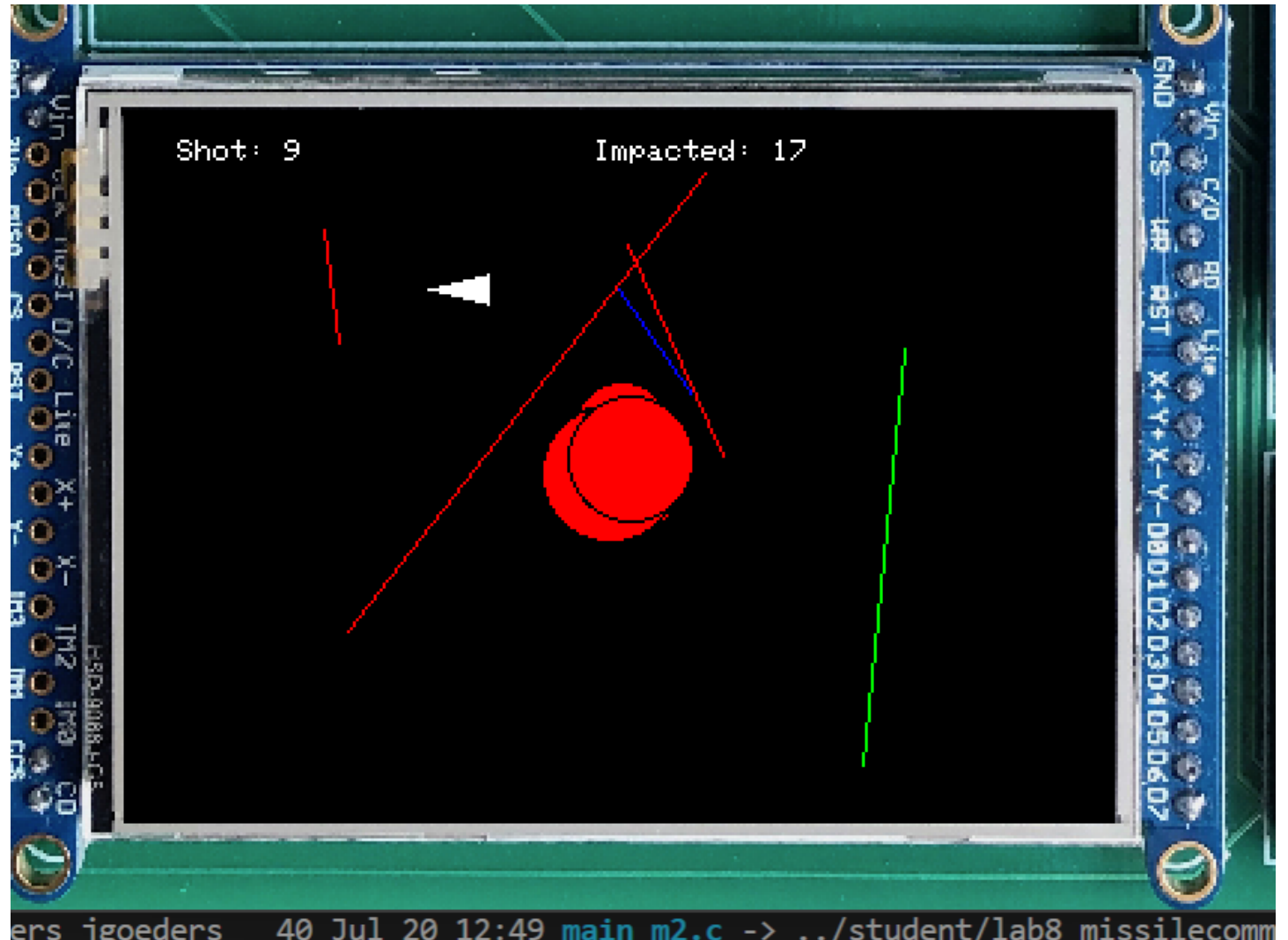
# Missile Properties (struct members)

type

- Player, Enemy or Plane
  - Green, Red or Blue

currentState

- You get to make your own state machine for the missiles.
- I made states for:
  - Initializing
  - Moving
  - Exploding Growing
  - Exploding Shrinking
  - Dead



x\_origin

y\_origin

x\_dest

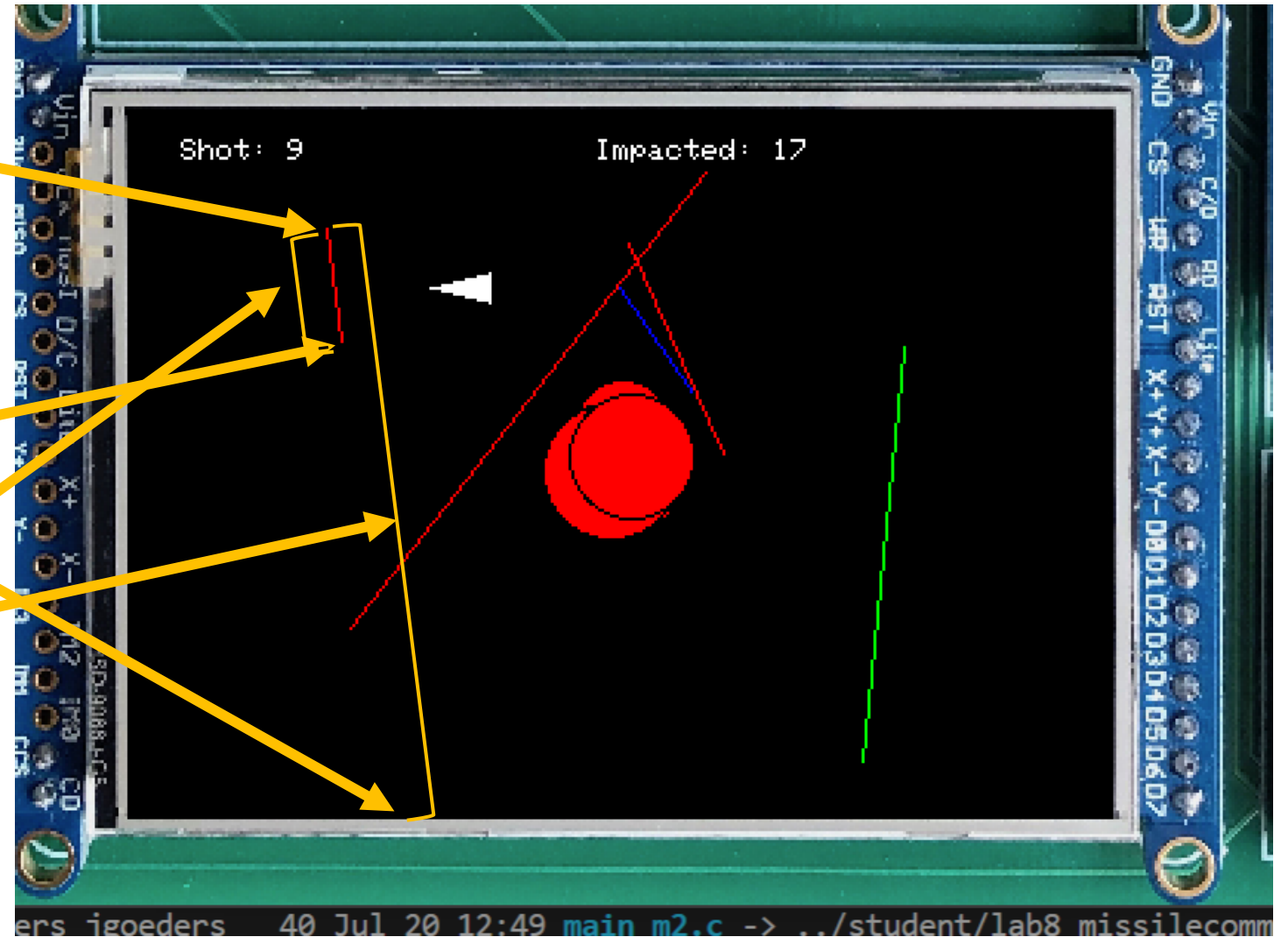
y\_dest

x\_current

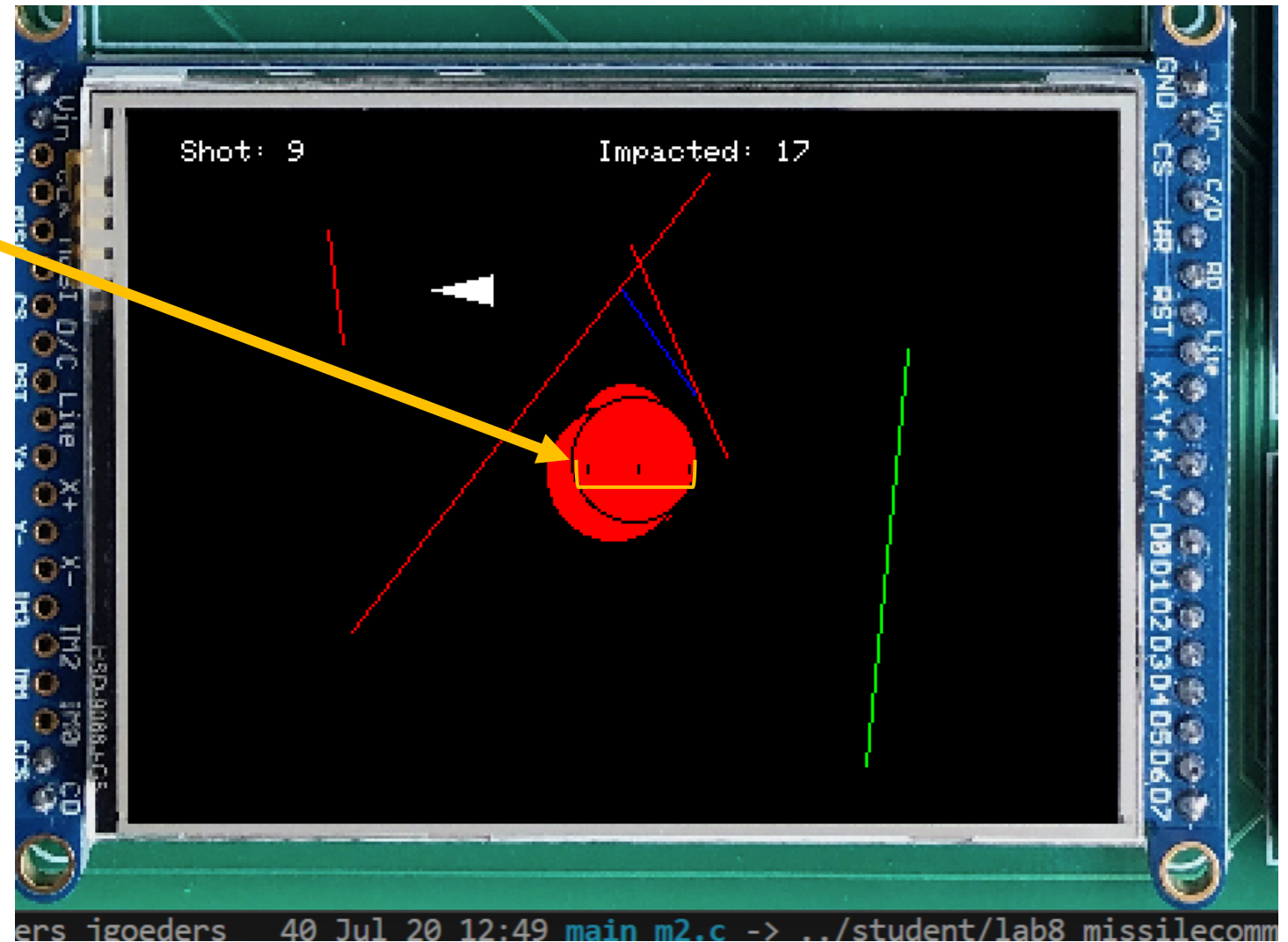
y\_current

total\_length

length



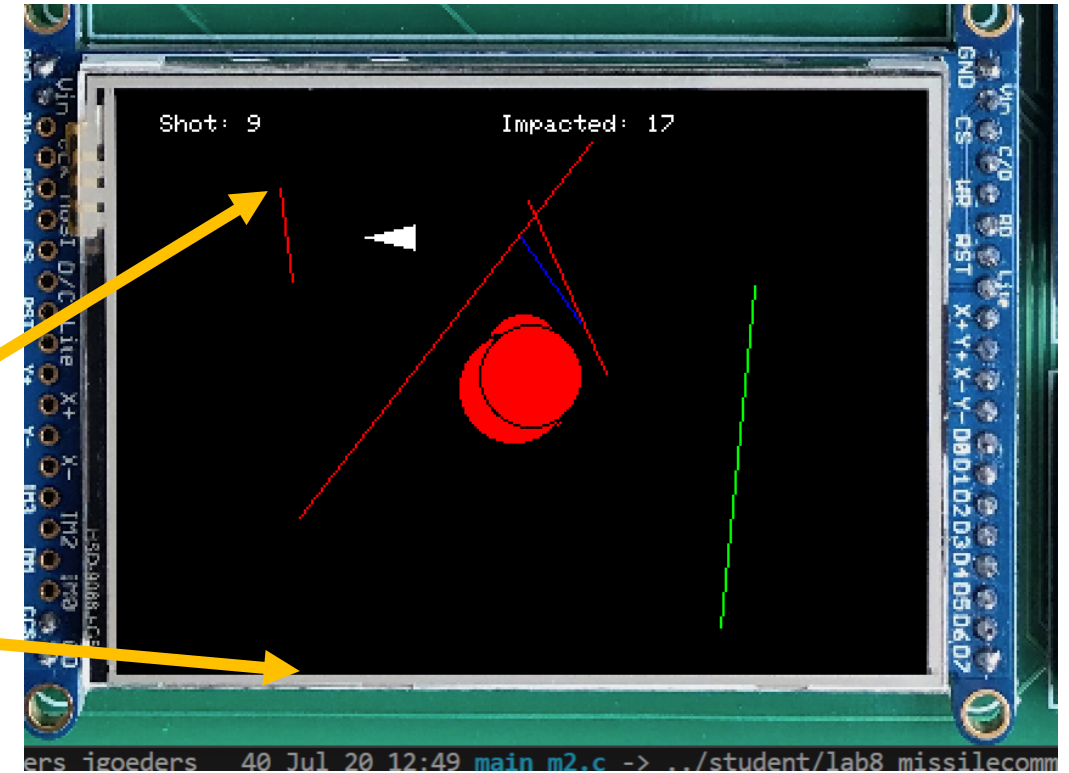
radius



# Initializing Enemy Missiles

```
#define CONFIG_MAX_ENEMY_MISSILES 7
```

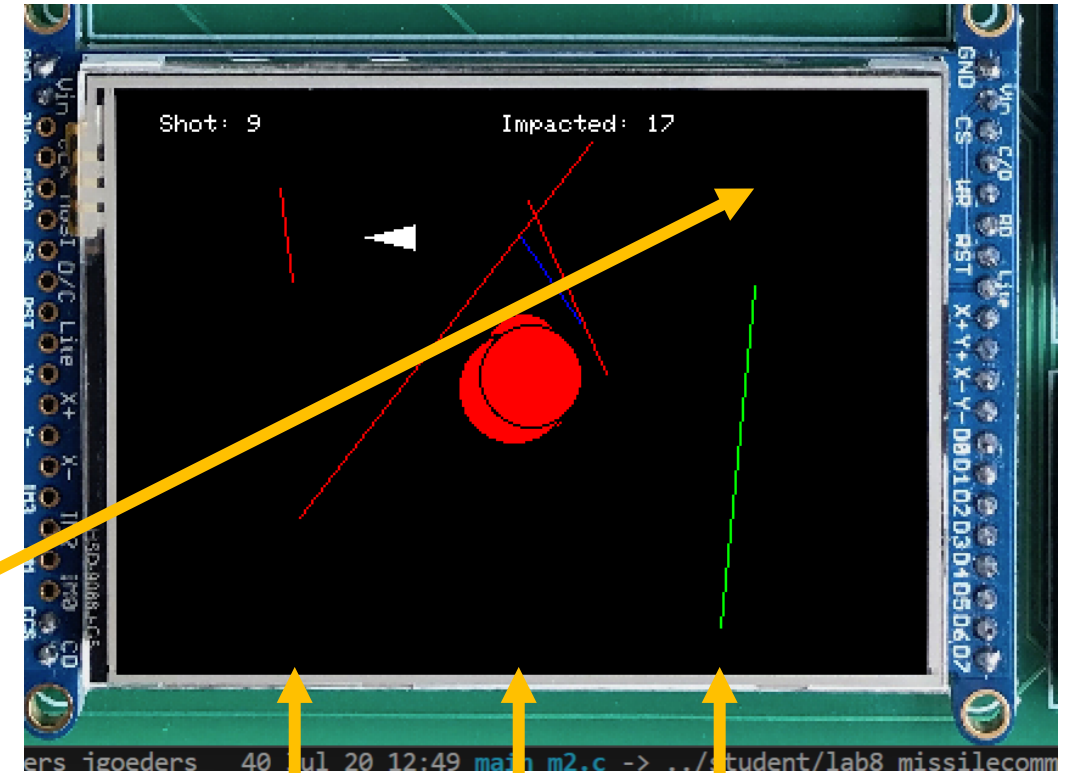
```
void missile_init_enemy(missile_t *missile) {  
    missile->type = MISSILE_TYPE_ENEMY;  
  
    // Set x,y origin to random place near the top  
    // of the screen (top quarter? - you choose!)  
  
    // Set x,y destination to random location along  
    // the bottom of the screen  
  
    // Set current state  
}
```



# Initializing Player Missiles

```
#define CONFIG_MAX_PLAYER_MISSILES 4
```

```
void missile_init_player(missile_t *missile, uint16_t  
x_dest, uint16_t y_dest) {  
  
    missile->type = MISSILE_TYPE_PLAYER;  
  
    // Set x,y origin to closest missile launch site  
  
    // x,y destination is provided (touched location)  
  
    // Set current state  
}
```



Missile launch sites (not drawn)  
3 sites spaced evenly on display

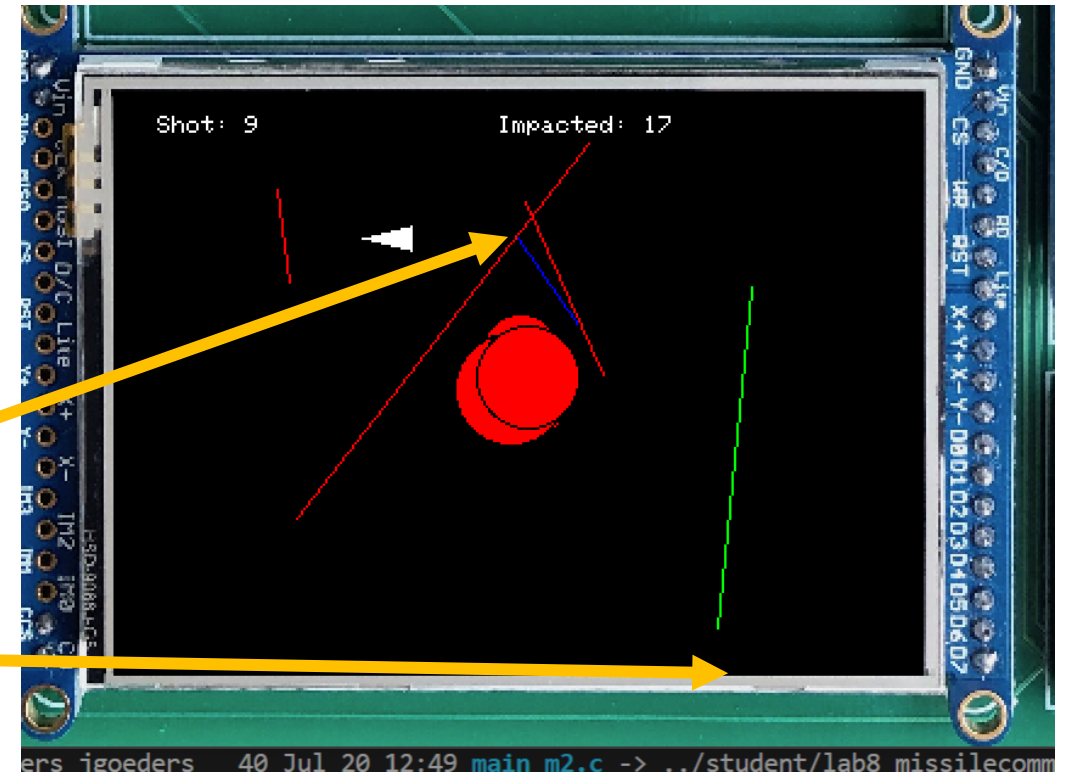


# Initializing Plane Missile (M3)

```
#define CONFIG_MAX_PLANE_MISSILES 1
```

You can assume this will never change

```
void missile_init_plane(missile_t *missile, int16_t  
plane_x, int16_t plane_y) {  
  
    missile->type = MISSILE_TYPE_PLANE;  
  
    // x,y origin provided (plane location)  
  
    // x,y destination chosen randomly along the bottom  
  
    // Set current state  
}
```



What should you do in your tick function?

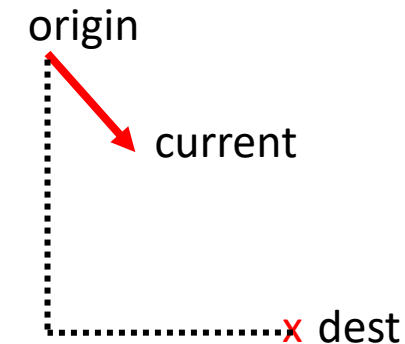
## If the missile is flying:

- Erase the old missile line
- Update length
- Calculate new current x,y
  - Calculate **percentage** of path traveled:
    - length / total\_length
- Draw the new missile line

```
// Speed of enemy missile
#define CONFIG_ENEMY_MISSILE_DISTANCE_PER_SECOND 40
#define CONFIG_ENEMY_MISSILE_DISTANCE_PER_TICK \
    (CONFIG_ENEMY_MISSILE_DISTANCE_PER_SECOND * CONFIG_TIMER_PERIOD)

// Speed of player missile
#define CONFIG_PLAYER_MISSILE_DISTANCE_PER_SECOND 350
#define CONFIG_PLAYER_MISSILE_DISTANCE_PER_TICK \
    (CONFIG_PLAYER_MISSILE_DISTANCE_PER_SECOND * CONFIG_TIMER_PERIOD)


// How fast explosion radius increases/decreases per second
#define CONFIG_EXPLOSION_RADIUS_CHANGE_PER_SECOND 30
#define CONFIG_EXPLOSION_RADIUS_CHANGE_PER_TICK \
    (CONFIG_EXPLOSION_RADIUS_CHANGE_PER_SECOND * CONFIG_TIMER_PERIOD)
```



$$x\_current = x\_origin + percentage * (x\_dest - x\_origin)$$

What should you do in your tick function?

**If the missile is exploding (increasing):**

- Increase radius
  - Draw circle
- 

```
// Speed of enemy missile
#define CONFIG_ENEMY_MISSILE_DISTANCE_PER_SECOND 40
#define CONFIG_ENEMY_MISSILE_DISTANCE_PER_TICK \
    (CONFIG_ENEMY_MISSILE_DISTANCE_PER_SECOND * CONFIG_TIMER_PERIOD)

// Speed of player missile
#define CONFIG_PLAYER_MISSILE_DISTANCE_PER_SECOND 350
#define CONFIG_PLAYER_MISSILE_DISTANCE_PER_TICK \
    (CONFIG_PLAYER_MISSILE_DISTANCE_PER_SECOND * CONFIG_TIMER_PERIOD)

// How fast explosion radius increases/decreases per second
#define CONFIG_EXPLOSION_RADIUS_CHANGE_PER_SECOND 30
#define CONFIG_EXPLOSION_RADIUS_CHANGE_PER_TICK \
    (CONFIG_EXPLOSION_RADIUS_CHANGE_PER_SECOND * CONFIG_TIMER_PERIOD)
```

**If the missile is exploding (decreasing):**

- Erase circle
- Decrease radius
- Draw circle

# Milestone 2: Game Control

In this milestone you will use your missiles to implement a basic version of the game.

gameControl.c:

- init()
- tick()

Need to handle:

- Launching enemy missiles (automatically)
- Launching player missiles (when screen touched)
- Detecting “collisions” and triggering explosions

**First check out main\_m1.c. It provided a very basic game control.**



Needs to keep an array of missile structs.

- You could use one big array, or separate arrays per type.

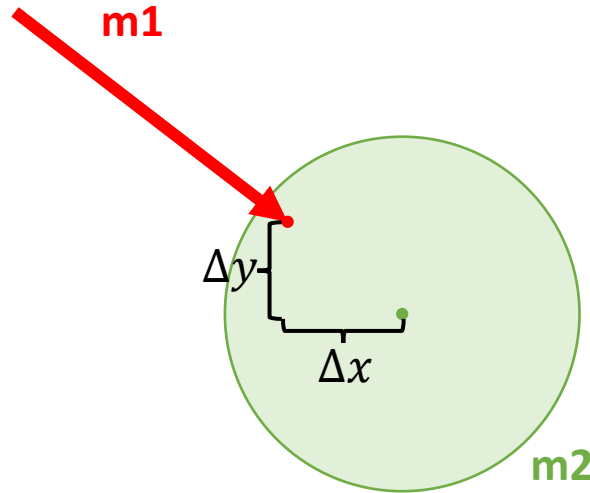
`gameControl_init()`

- Initialize all of your missiles

`gameControl_tick()`

- Tick all of your missiles
- If enemy missile is dead, relaunch it (call init again)
- If touchscreen touched, launch player missile (if one is available)
- Detect collisions

# Detecting Collisions



Check if m1 is inside m2:

$$\Delta y^2 + \Delta x^2 < radius^2$$

- Any explosion type blows up enemy/plane missile
- Player missiles only explode when they reach their destination.

## How to trigger an explosion?

- Set the `explode_me` struct member to true, and make sure your state machine checks this while the missile is flying.

You may not have enough time to tick all of your missiles!

- Take a while to draw/erase explosions.
- If lots of things are exploding you will miss interrupts and the game will slow down

**What could you do?**

Tick half of the missiles each `gameControl_tick()`.

- If you are ticking half as often, make sure you move/resize objects twice as fast.

# Game Control

Needs to keep an array of missile structs.

- You could use one big array, or separate arrays per type.

gameControl\_init()

- Initialize all of your missiles

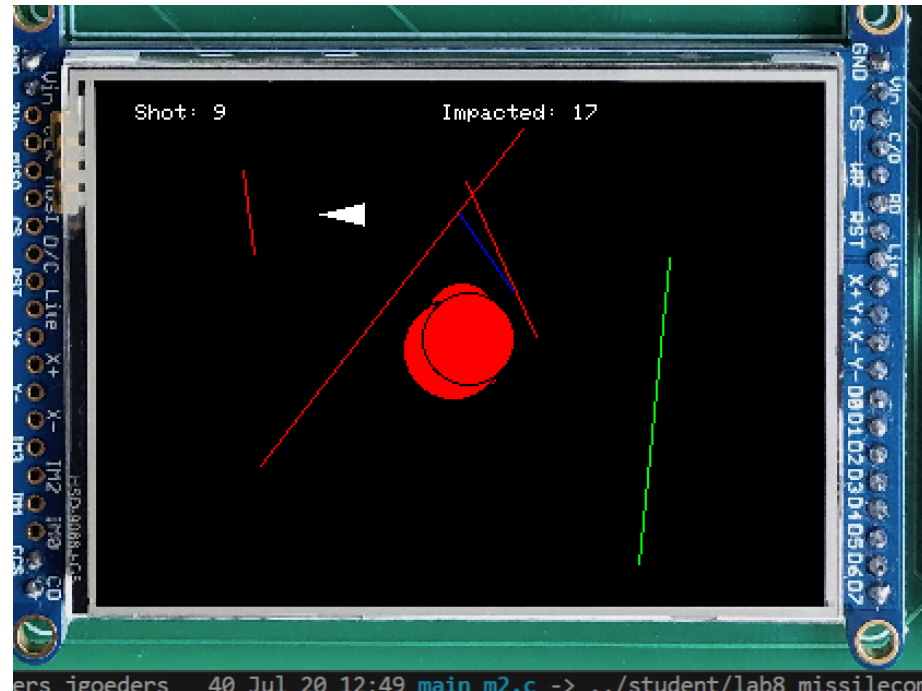
gameControl\_tick()

- Tick **HALF** your missiles
- If enemy missile is dead, relaunch it (call init again)
- If touchscreen touched, launch player missile (if one is available)
- Detect collisions



# Milestone 3: Plane + Stats

- Create a state machine for your plane
  - Fly right to left
  - Plane can be destroyed by an explosion (no explosion animation)
  - Shoot a missile while flying
  - Wait a while to reappear after leaving or being destroyed



- Keep track of
  1. Number of missiles shot by player
  2. Number of enemy/plane missiles that impact the ground
    - The **impacted** struct member can be used for this
      - Set to true when impacted
      - Set to false when you read it and update your stat counters
- Draw the stats at the top of the screen

# Updated Game Control for M3

gameControl\_init()

- Initialize all of your missiles
- **Initialize stats**

gameControl\_tick()

- Tick HALF of your missiles
- **Tick plane**
- If enemy missile is dead, relaunch it (call init again)
- If touchscreen touched, launch player missile (if one is available)
- Detect collisions
- **Draw stats**