OVERVIEW:
+ Do about 1 week in, after students have done tutorial
+ Post AFTER class

GOAL: help implement Project

BUT ALSO --> Practical issues with real-time, distributed state simulation! (i.e., also CLASS MATERIAL)

0) ASSUME

Tutorial done

-- Worked through! Not just compiled game-final.zip

Networking is complete and debugged

-- NetworkManager, EventNetwork and Sentry

-- Tested! Be sure to test thoroughly before proceeding!

1) PICTURE

```
B E            Objects
S S H  HOST   --------->     CLIENT
NM SENTRY   <--------       NM SENTRY
              P2 Input
              DF
```

2) QUESTION - why is synchronization needed? If each PC runs the exact same simulation, no need to synchronize!

- This means exact same random seed, too!

NOTE: Random number generation is complex function. e.g.,

```
// Generate "random" number.
int rand()
  g_next = ((5 * g_next) + 1) mod 16

// "Seed"
void srand(int seed)
  g_next = seed
```

e.g., Host picks from rand() - 32, 12, 10, 64 ...
Client picks from rand() - 32, 12, 10, 64 ...

But if Objects processed in slightly different order (e.g., Saucer 1 before Saucer 2), will be off!

- User input adds variability --> takes time to propagate to other host

Even if Objects stored in same order, latency from event (e.g., user input) could mean additional random number drawn --> will be off!
-- Will be out of sync during travel (and could be some effect)
- So *cannot* be done at the exact same time
  = Could apply "timestamp" and either:
    + Delay user action - would feel like lag. How long to delay?
    + Roll back Host state (time warp) - complicated and change what host sees

3) THEREFORE -> HOST is authoritative.
   -- Has the final say of the world.
   -- CLIENT will simulate as much as possible, but HOST is responsible for "important" decisions (e.g., is Hero hit by Saucer?)
   (Note: has side benefit of helping prevent cheating by CLIENT)

4) QUESTION: What player input commands does CLIENT send?
   KEY - for keystrokes
   MOUSE - for when mouse is clicked
   Note: Do *not* need to send when Mouse is moved. Do not need to show opponent’s RETICLE.
   KEY includes keypressed and MOUSE includes (x,y)
   Note: Client can check for valid key before sending (e.g., no need to send non-recognized keystroke)

5) QUESTION: What object commands does HOST send?
   NEW - whenever a new object is created
   UPDATE - whenever an existing object has changed
   DELETE - whenever an object is destroyed
   Each includes Object ID, and UPDATE and DELETE includes serialized attributes
   e.g., CLIENT receives DELETE

   df::WorldManager &world_manager = df::WorldManager::getInstance();
   Object *p_obj;
   p_obj = world_manager.objectWithId(id);
   if (p_obj == NULL)
     // error! not found
     world_manager.markForDelete(p_obj);

6) QUESTION: What are all the game Objects for Saucer Shoot 2?
   Bullet
   Explosion
   Hero
   Points
   Saucer
   Score
   Stars
   GameStart (not required)
7) QUESTION: Do all need to be synchronized?

STARS --> QUESTION: does it matter if they deviated in location/speed?
-- Could send NEW when HOST creates
-- Could have both HOST and CLIENT create their own "set" upon startup

SAUCER
-- Position matters --> send NEW
-- Client and Host can both do velocity (no need to update position)
-- QUESTION: when would they deviate?
  ANSWER: when "respawns" in random location off to right --> UPDATE
-- QUESTION: what about animations?
  ANSWER: never need to synchronize (a "decoration")

COLLISION
-- Both Client and Host can simulate collision
-- But Host needs to officially determine outcome (authoritative)
-- Destroy Bullet and Saucer --> DELETE
  QUESTION: What about EXPLOSION?
  -- Could create on Host and send NEW .... QUESTION: or ...?
  -- Host and Client both create when Saucer dies (saves bwidth)

HERO
-- Does not have velocity
-- When Player 1 key --> Host moves --> UPDATE
-- When Player 2 key --> Client could move --> UPDATE to Host
  QUESTION: But what if move was invalid (e.g., Saucer there or Hero there)?
  ANSWER: Host would tell Client, and Client "rollback" / "fix" --> BLEAH
  QUESTION: So, why would a system ever do that?
  ANSWER: Avoid LAG. Basically, otherwise at least 1 RTT for response
-- SO, when Player 2 key
  --> Client sends KEY
RETICLE

-- QUESTION: Does opponent care where this is? Probably not.

--> Don’t synchronize

-- Host mouse click --> new Bullet --> NEW

-- Client mouse click

--> send MOUSE (x,y)

--> Host receives, creates Bullet --> NEW

-- NOTE: can do "smart" checking on Client

e.g., when click, is too soon to spawn --> if so, no need to send

BUT --> Host will still need to check, too --> avoid CHEAT

POINTS

-- When change value --> UPDATE

-- Could do "time" / "ticks" locally, so only UPDATE when Saucer destroy

8) QUESTION: How to "detect" changes in HOST?

Host poll all Objects every step

Could just serialize() every object

--> remember, only sends changes since last serialize()

QUESTION: why not?

ANSWER: even "decoration" changes serialized

--> e.g., animation

So, check specific attributes --> isModified()

--> e.g., isModified(df::POS)

Send as appropriate

TIP: Make function, bool needSynch(Object *p) --> TRUE if synch, else FALSE

// Bullet synchronized when created
if (p_o->getType() == "Bullet-Host" || p_o->getType() == "Bullet-Client") {
    if (p_o->isModified(df::ID))
        return true;
    return false;
}

// Hero synchronized when moves or is created.  SHOW DIFF ONLY
if (p_o->getType() == "Hero-Host" || p_o->getType() == "Hero-Client") {
    if (p_o->isModified(df::ID) || p_o->isModified(df::POS))
        return true;
    return false;
}

... 

NOTE -> SAUCER (additional force synch in move-to-start)

// Saucer only synchronized when created.  
// Movement handled locally (synchronized again in moveToStart()).
if (p_o->getType() == "Saucer") {
    if (p_o->isModified(df::ID))
        return true;
    return false;
}

USE IT!!

// Only send objects needing synchronization.
df::ObjectList all_objects = world_manager.getAllObjects();
df::ObjectListIterator i(&all_objects);
for (i.first(); !i.isDone(); i.next()) {
    if (needSynch(p_o)) {
        // Set message type.
        // If object id is modified, assume NEW
        HostMessageType msg_type;
        if (p_temp_o->isModified(df::ID))
            msg_type = ADD_OBJECT;
        else
            msg_type = UPDATE_OBJECT;

        sendObject(p_temp_o, msg_type);
    }
}

9) NOTE - needs player-versions of some objects

-- HOST-HERO and CLIENT-HERO.  QUESTION: Others?
    -- Bullets (color and who gets points)
    -- Points
    -- Could make separate Object, but duplicate a lot of code.
    QUESTION: Alternative?
-- Could create "bool is_host" functionality.  Act appropriately.

Hero get keyboard input
if isHost()
    // apply to Host-Hero
else
    // send to Host
end if
Use ROLE singleton (see slides/writeup)

10) REMEMBER Saucer Shoot 2 only needs

Core gameplay

Does not need:

    GameStart
    Nuke Display
   GameOver

HOST starts - waits for CLIENT

CLIENT connects

--> Start moving and shooting!

When either/both die

--> Game exits (gracefully)

NOTE: Can add extras for 5% Misc points

-- If so, Subtle - GameStart is "inactive()" --> getAllObjects(true)

HAPPY SHOOTING!