# Smart Cameras

Mark DiVelbiss, Selena Grant, Qing Liu

# Overview

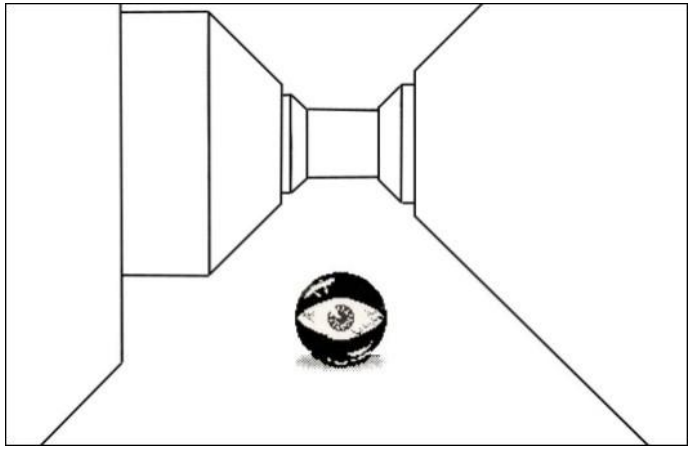
- First Person vs Third Person

- Common Problems

- Motivations

# First Person vs Third Person Camera

- First Person
  - Immersive
  - Intuitive



- Third Person
  - “Abstract” POV
  - More knowledge available to the player

# Third Person - Fixed Cameras

- Fixed Position
  - Camera moves from position to position as needed
  - No dynamic behavior
- Fixed Angle
  - Tracking based
  - Often used for aerial views


# Third Person - Dynamic Cameras

- Player Controlled
  - Mostly or completely the player's job to control
  - Can distract from gameplay and reduce immersion
- AI Controlled with Player Interruption Allowed
  - Primarily AI responsibility
  - Player can impose action
  - "Gamatography"

# Common Problems

- Focusing on the player rather than the goal

- Balancing player agency with camera AI

- Small Environments

# Motivations

- The player’s responsibility?

- What information should the player have access to?

- Does it add to or limit the game?


- What makes sense?

# Considerations
# &
# Application

# Design Choices

We know types of cameras and what they're good for, but...

- What makes a camera smart?
- Can I have multiple types?
- What can go wrong?
- Is the camera player controlled?

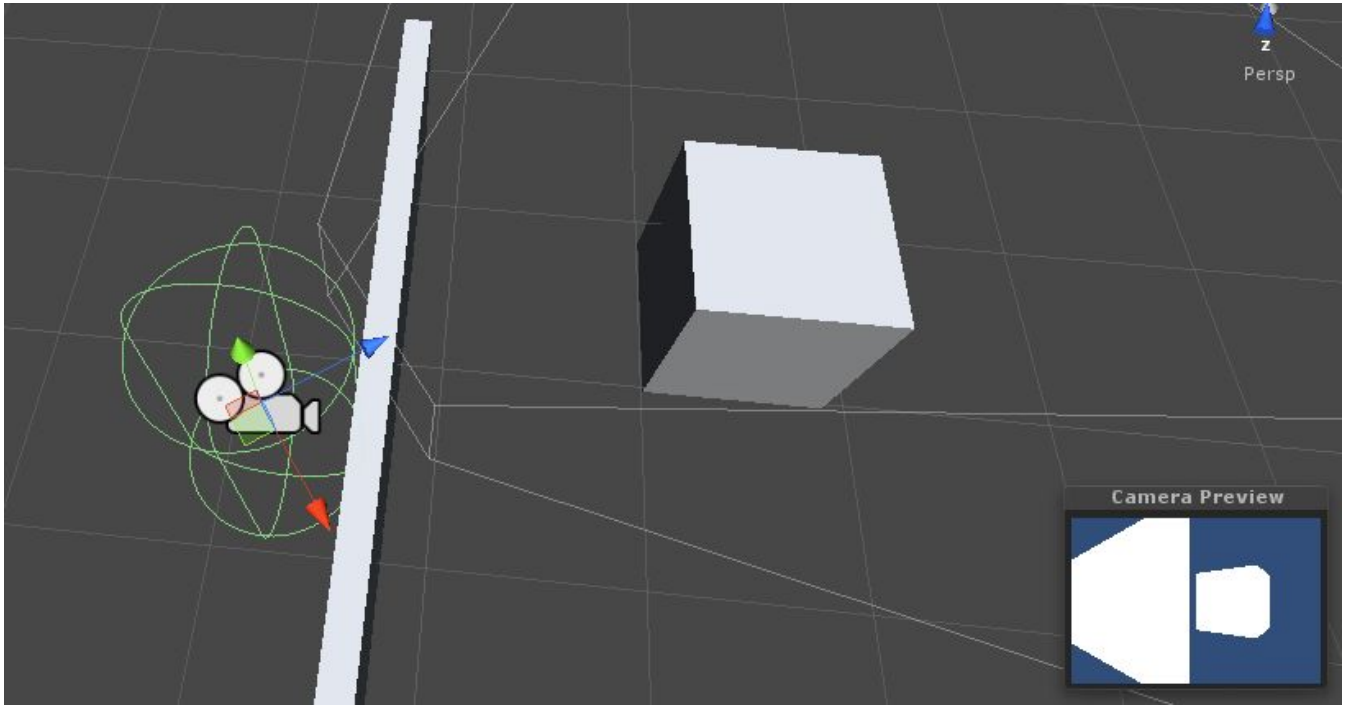
# Clipping and Flipping

Clipping: The camera passes through geometry.

Flipping: Camera can’t find a resting point.

# Solutions?

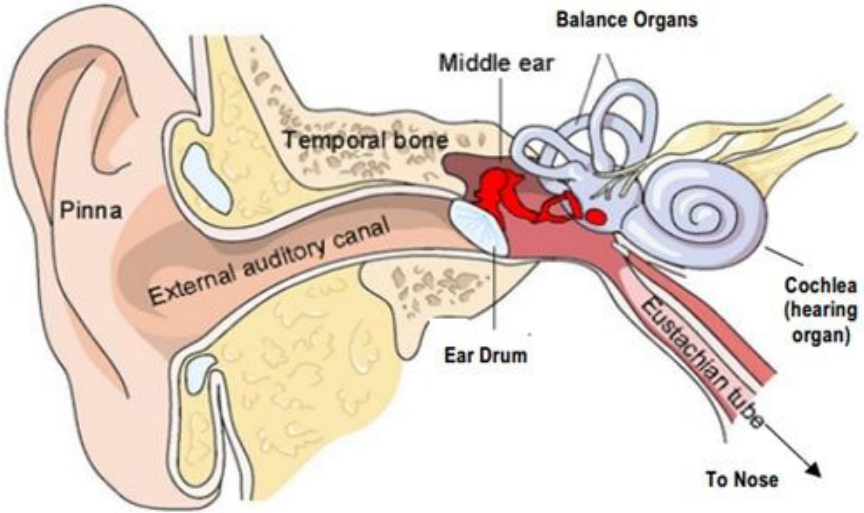
- More Raycasts / Better Predictions
- Adjust Near Plane / Frustum
- Avoid Corners
- Have camera “scoot” up walls



# Doom Induced Motion Sickness (DIMS)

Yes this is a real thing.

Motion sickness caused by the perception of movement when the inner ear detects none.




MakeAGIF.com

# Causes?

- Screen Bobbing
- Bright Screen/Room
- Bright/Unrealistic Colors
- Small field of view
- Being too close to the screen
- First-person camera
- New to video games

# Solutions?

- Fixed object to focus on.
- Flat ground
- Dim your screen
- Turn off head bobbing
- Widen FOV
- Don’t drink caffeine?

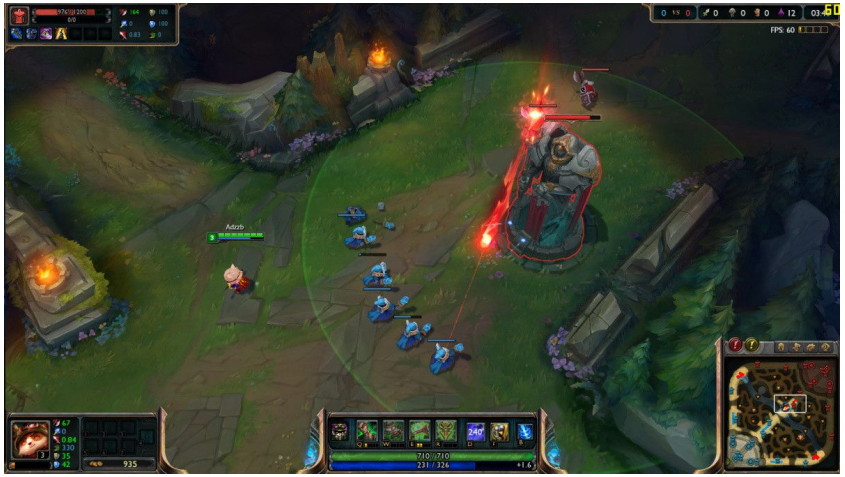
Don’t worry! You’ll acclimate over time.

# Camera Control

Tracking vs. Pushable

Movable vs. Floating

How much agency do we give the player to determine their view?



# Restricting Camera Control

Benefits

1. Fewer bugs (bugs break immersion)
2. More design options
3. More visually appealing
4. Information control
5. Less of a hassle

# Camera Control Freedom

Benefits

1. Increased Engagement/Immersion
2. Finer Detail
3. Encourages Exploration
4. Smart Freedom
    a. Reorientation Button
    b. Player never leaves screen

# Transition

Why not have multiple camera systems?

You can, but “jump cuts” will disorient and confuse players.

What you need is a transition, a way to communicate to the player through the game that their view is changing.

# Examples

1. Smooth camera motion
2. Allow the player a brief pause
3. Animate the change

# Implementation

# Overview

- Camera smart follow implementations
  - Auto avoid occlusion
  - Make occlusion transparent
- Maintain multiple objects in camera
  - Enemy lock on function

# How does camera follow in game?

- Third person follow
  - Obstacles may block the sight between camera & player

- Auto avoid
  - Zooming
  - Rotating

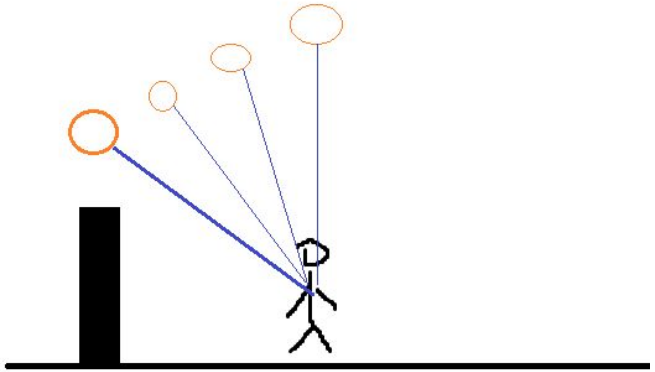
- Make transparent

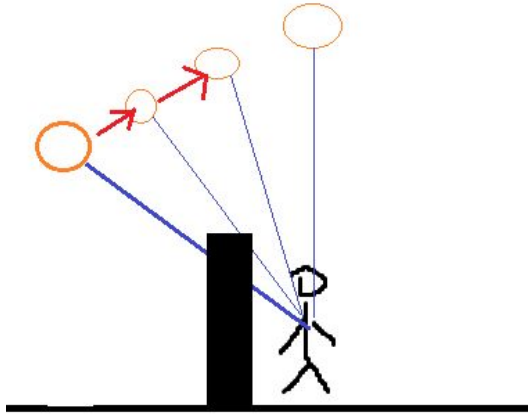
# Camera follow implementations

- Auto avoiding by rotating

- Demo

# Camera follow implementations





# Camera follow implementations

```
// The first is the standard position of the camera.
checkPoints[0] = standardPos;
// The next three are 25%, 50% and 75% of the distance between the standard position and abovePos.
checkPoints[1] = Vector3.Lerp(standardPos, abovePos, 0.25f);
checkPoints[2] = Vector3.Lerp(standardPos, abovePos, 0.5f);
checkPoints[3] = Vector3.Lerp(standardPos, abovePos, 0.75f);
// The last is the abovePos.
checkPoints[4] = abovePos;
// Run through the check points...
for(int i = 0; i < checkPoints.Length; i++)
{
    // ... if the camera can see the player...
    if(ViewingPosCheck(checkPoints[i]))
        // ... break from the loop.
        break;
}
// Lerp the camera's position between it's current position and it's new position.
transform.position = Vector3.Lerp(transform.position, newPos, smooth * Time.deltaTime);
```

# Camera follow implementations

```
bool ViewingPosCheck (Vector3 checkPos)
{
    RaycastHit hit;

    // If a raycast from the check position to the player hits something...
    if(Physics.Raycast(checkPos, player.position - checkPos, out hit, relCameraPosMag))
        // ... if it is not the player...
    if(hit.transform != player)
        // This position isn't appropriate.
        return false;

    // If we haven't hit anything or we've hit the player, this is an appropriate position.
    newPos = checkPos;
    return true;
}
```

# Camera follow implementations

- Make transparent

- Demo

# Camera follow implementations

- Find the occlusions between Camera and player

- Apply transparent function

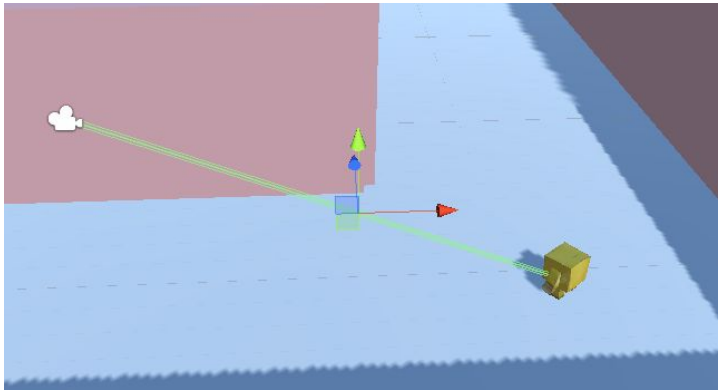
- Run restore functions when they are no longer blocking

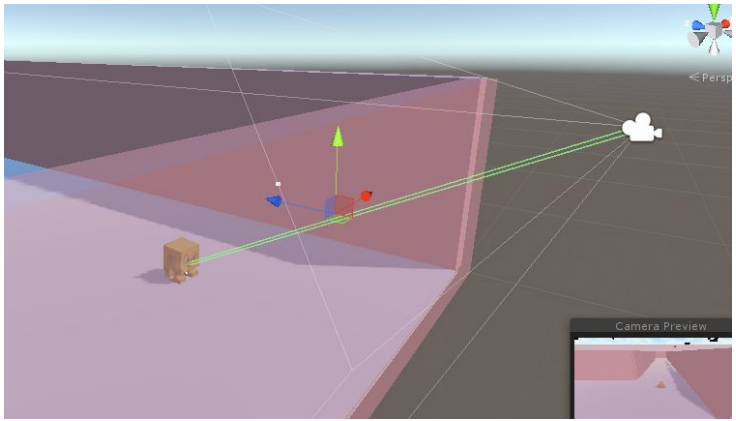
# Camera follow implementations

- Two approaches
  - Ray casting
    - Easy to find out the obstacles
    - Hard to know when to restore
  - Collision detection
    - Need one collider
    - Easy to know when to restore

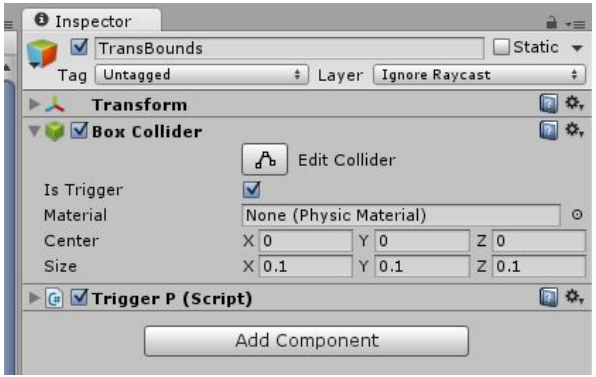
- Collision detection ✓

# Camera follow implementations









# Camera follow implementations

```
void OnTriggerEnter (Collider other)
{
    WallScript isWall = other.gameObject.GetComponent<WallScript>();
    if (isWall) {
        isWall.TransActive ();
    }
}

void OnTriggerExit (Collider other){
    WallScript isWall = other.gameObject.GetComponent<WallScript>();
    if (isWall)
        isWall.TransInActive ();
}

void LateUpdate()
{
    mag = (this.transform.position - player.transform.position).magnitude;
    bc.size = new Vector3 (bc.size.x, bc.size.y, mag);
    bc.center = new Vector3 (bc.center.x, bc.center.y, mag/2);
}
```

# Multiple objects in camera

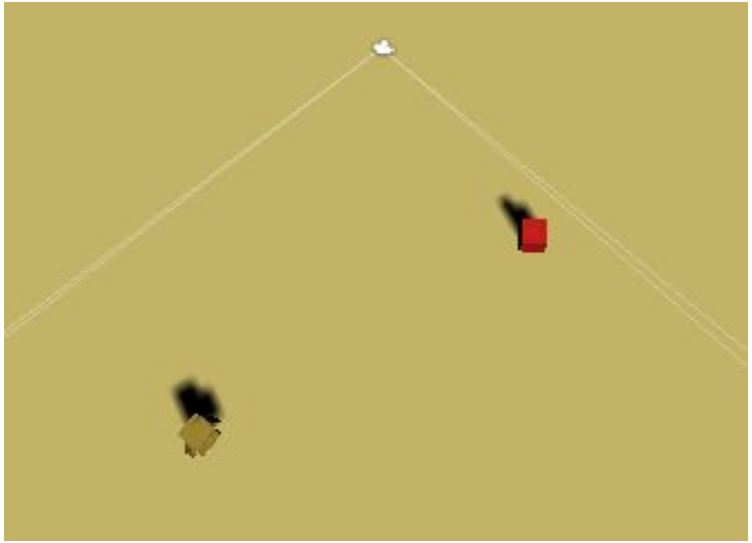
- Multiple objects
  - Focus on something
  - Enemy lock on
  - etc.

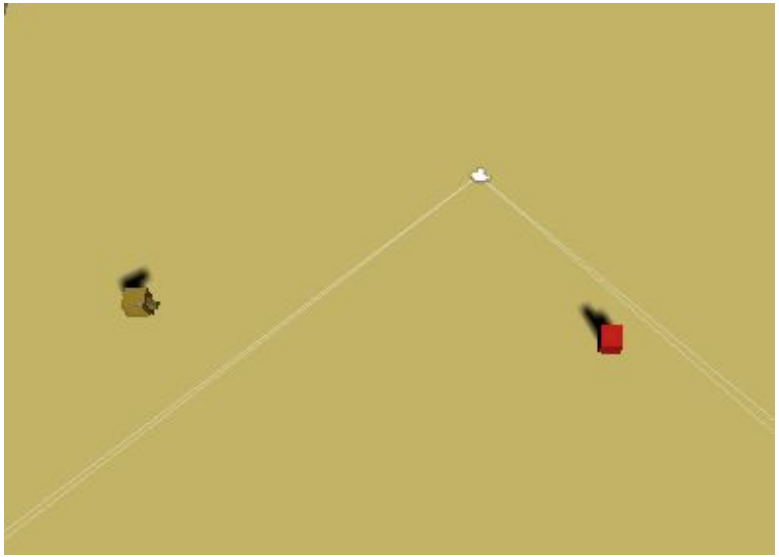
# Multiple objects in camera

- Enemy lock on function

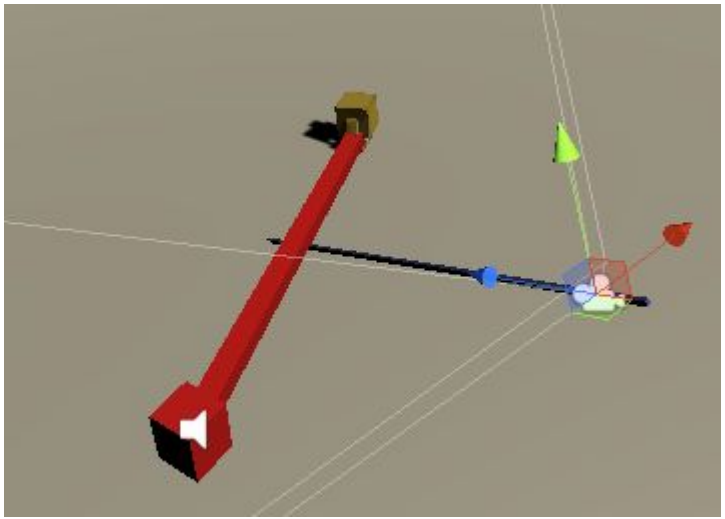
- Demo

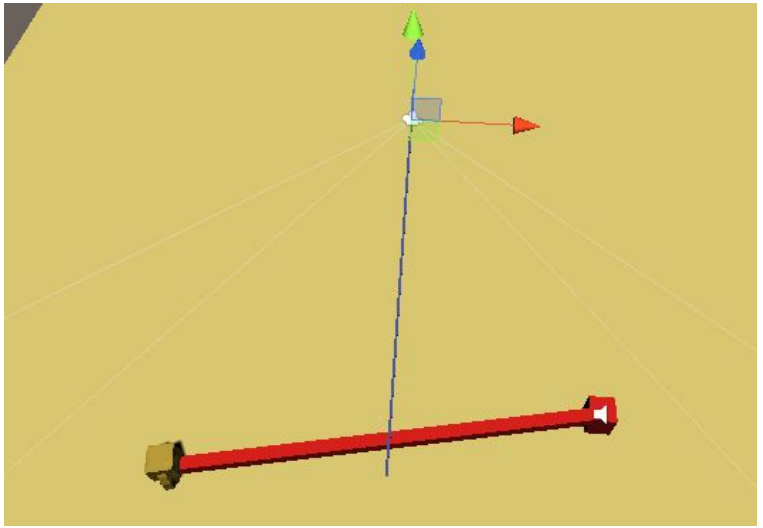
# Multiple objects in camera



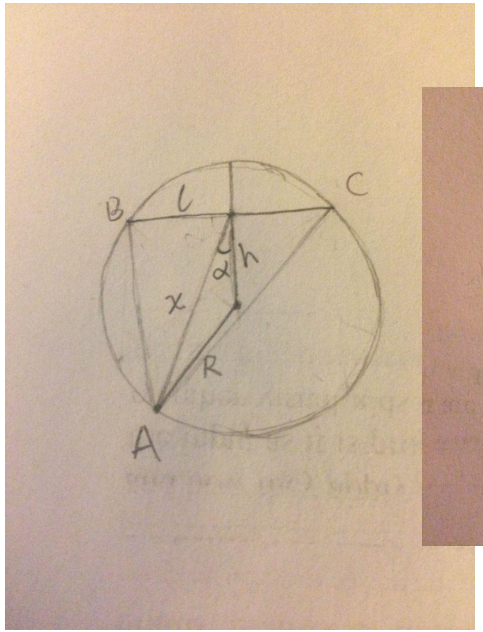


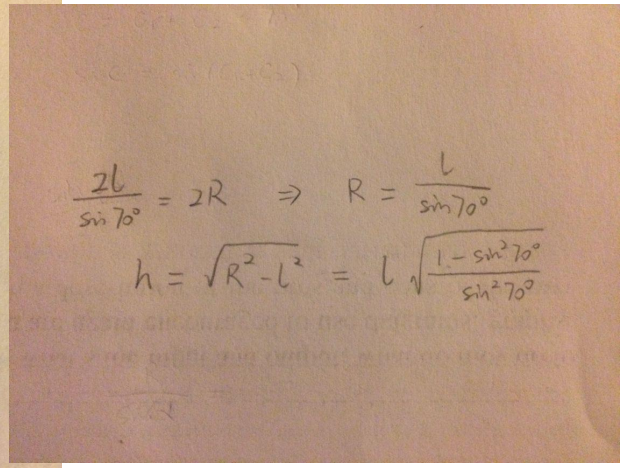
# Multiple objects in camera





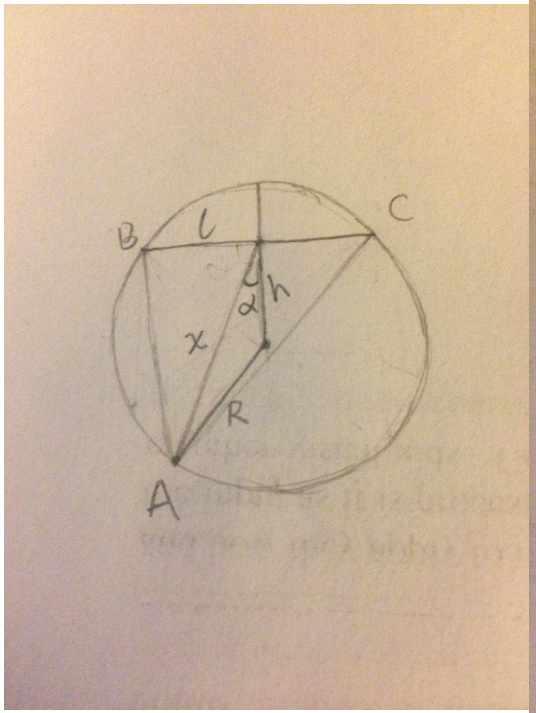
# Multiple objects in camera

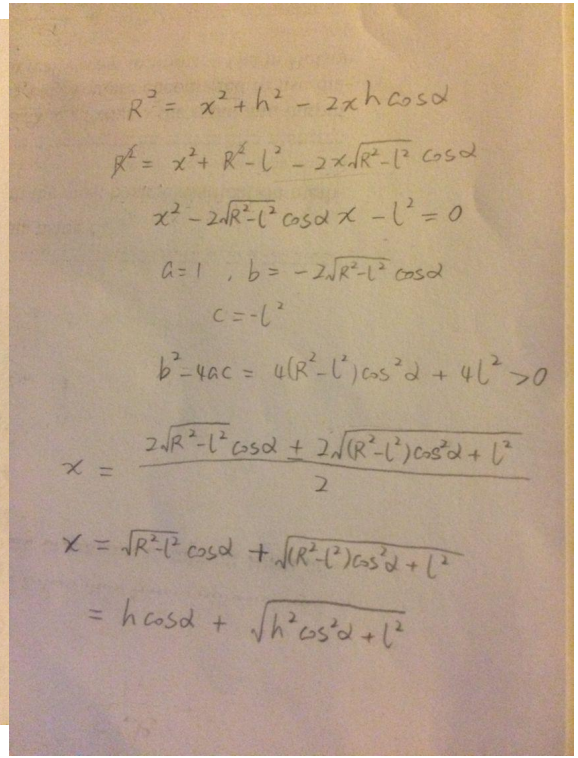



- Circumscribed circle
- Law of sine
- Pythagorean theorem

# Multiple objects in camera




- Law of cosine

# Multiple objects in camera

```
if (Input.GetKeyDown (KeyCode.K)) {
    if (lockOnEnemy != Enemie1) {
        lockOnEnemy = Enemie1;
    } else {
        lockOnEnemy = null;
        targetOffset = originOffset;
    }
}

 if (lockOnEnemy)
     target.position = (player.position + lockOnEnemy.transform.position) / 2;
 else
     target.position = player.position;


if (lockOnEnemy) {
    float zoomdist = CalZoomDist ();
    targetOffset = originOffset.normalized * zoomdist;
    if (targetOffset.magnitude < originOffset.magnitude)
        targetOffset = originOffset;
}
```

# Questions?

# Work cited


GDC 2014 Talk - 50 Game Camera Mistakes: https://www.youtube.com/watch?v=C7307qRmlMI

Extra Credits. “Simulation Sickness - Causes and Cures for Game Headaches - Extra Credits.” Online Video Clip. Youtube. Youtube, 19 Feb 2014. Web. 31 Mar 2016.
Haigh-Hutchinson, Mark. "The Camera as an A.I. Game Object."*Gamasutra*. N.p., n.d. Web. 01 Apr. 2016.
Kelly, Tadhg. "Camera Comes First [Game Design]." *'What Games Are'*N.p., 25 Oct. 2011. Web. 01 Apr. 2016.
Rogers, Scott. *Level Up!: The Guide to Great Video Game Design*. Chichester: Wiley, 2010. Internet resource.

Pictures:
https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwj0zITNh-rLAhULGR4KHZ74AIEQjRwIBw&url=https%3A%2F%2Fplay.google.com%2Fstore%2Fapps%2Fdetails%3Fid%3Dcom.ea.game.pvzfree_row&bvm=bv.118443451,d.dmo&psig=AFQjCNG9462bbNqCD77Iz1h1-7vjMRaXtA&ust=1459483865586013
https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwjk28-AiOrLAhXGKh4KHTwxA1sQjRwIBw&url=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3D3AgcKKjxDME&bvm=bv.118443451,d.dmo&psig=AFQjCNFJAlAtg-GVfcfn3XVcGaReiu_IhQ&ust=1459483925550428
https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwiw-NSPlOrLAhVDXh4KHXc_DrgQjRwIBw&url=http%3A%2F%2Fwww.playstationlifestyle.net%2F2015%2F04%2F04%2Fdevil-may-cry-4-special-edition-tweaks%2F&bvm=bv.118443451,d.dmo&psig=AFQjCNEEZq6yP6j2bcxHgFhdxoWtQwN5IQ&ust=1459487314412816

Website:
http://unity3d.com/cn/learn/tutorials/projects/stealth/camera-movement?playlist=17168

Tools:
Physics platformer kit from Buckeye box