DEPARTMENT OF
INFORMATION SYSTEMS
AND COMPUTER SCIENCE

Platformer Camera

Looks Hard

Lecture Time!

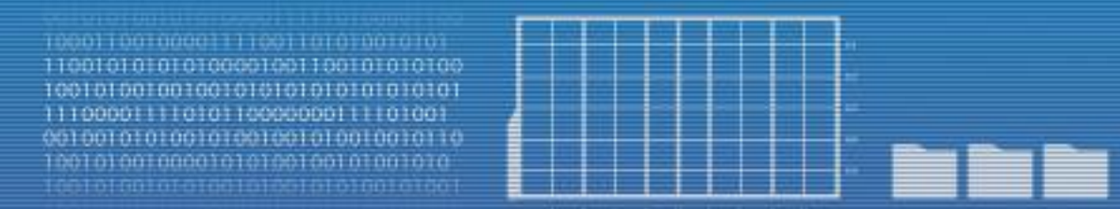
- Homework: A Preview
- Viewport: Controlling What Gets Drawn
- Camera: Tracking (or Not)
- Homework: Specs

Setting a Camera and Viewport

```
// SFML's struct for viewport
sf::View view;

// what camera should focus on,
// in pixel coordinates
view.setCenter( someVector2f );

// size of area to be shown
// by camera, in pixels
view.setSize( anotherVector2f );
```

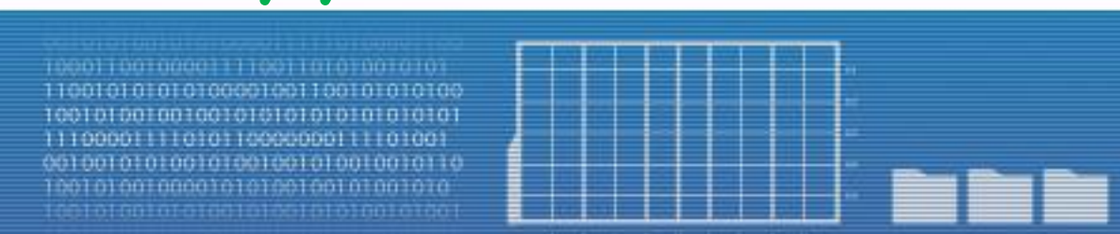
Setting a Camera and Viewport

```
// actual area of window that camera
// will render to, in percentage

// entire window
view.setViewport( sf::FloatRect(
   0, 0, 1, 1 ) );

// centered,
// half of window dimensions
view.setViewport( sf::FloatRect(
   0.25f, 0.25f, 0.5f, 0.5f ) );
```



Setting a Camera and Viewport

```
// actually setting the camera
window.setView( view );

// draw stuff here

// note that any change to view
// will not have any effect
// until you call setView() again

// note: window is your
// sf::RenderWindow variable
```



DISCS

Setting a Camera and Viewport

```
// reset to default view
window.setView(
   window.getDefaultView() );

// draw GUI stuff here
```

DISCS

Lights, ____, Action!

- Games with worlds larger than the screen will live or die by how appropriate the camera is for the game and by how well the camera's implementation is
- The faster the game's pace, the stricter the requirements become
- And it's not just about tracking a point of interest

Lights, ____, Action!

- Motion sickness is very much a threat, even in 2D games
- Caused by conflicting sensory signals
  - Body feels acceleration but eyes don't catch signs of movement
  - Eyes pick up changes in background but no actual inertia occurs
  - Would also explain why some people can't read in a moving vehicle without experiencing nausea

Power of Three

- There are three challenges when implementing scrolling
  - Attention
  - Interaction
  - Comfort

Attention

- Use the camera to provide sufficient game information and feedback
  - What the player needs to see

Little Witch Academia, episode 1

Interaction

- Player should have clear control over what's displayed
- Make background changes predictable and tightly bound to controls
  - What the player wants to see

Akiba's Trip: The Animation, episode 11

Comfort

- Ease and contextualize background changes
  - How to reconcile needs and wants smoothly and comfortably

Miss Kobayashi’s Maid Dragon, episode 11

Position-Locking

- Camera should focus on the character that the player is controlling
- Simple to implement but still very useful
- Camera tends to make many unnecessary movements
  - ANY movement by the player character, no matter how small, will result in the camera moving as well

Edge-Snapping

- Sometimes the camera should not be allowed to scroll past a specific point
- Assuming a rectangular world, camera position can be restricted to some distance away from the world's edges to prevent rendering unnecessary areas

Camera-Window

- Camera can be made stationary until player character hits the edges of a predefined box within the window
- Player character effectively pushes against these edges to move the camera

Camera-Window

- Large boxes prevent most unnecessary camera movement but also prevent the player from seeing what's ahead
- Small boxes allow the player to see what's ahead but also increases the likelihood of unnecessary camera movements

Position-Snapping

- To allow the player to be able to look ahead, the camera-window can be combined with a slower but continuous alignment of the camera
- This minimizes rapid camera motion and still allows the player to focus on what's ahead

Platform-Snapping

- This camera technique is somewhat unique to platformers as it relies on the guarantee that the player character will eventually land on a platform
- The camera-window technique is still used, but the camera will drift to the player when s/he lands on a platform

Homework

- Modify your previous platformer homework to implement a camera
- Add three new properties that will define the camera's behavior

DISCS

Homework

```
H_ACCEL         10     // per-second
H_COEFF         0.3    // N/A
H_OPPOSITE      2.0    // N/A
H_AIR           1.0    // N/A
MIN_H_VEL       0.01   // per-frame
MAX_H_VEL       200    // per-second
GRAVITY         20     // per-second
V_ACCEL         -600   // per-second
V_HOLD          1      // N/A
V_SAFE          6      // N/A
CUT_V_VEL       -20    // per-second
MAX_V_VEL       400    // per-second
GAP             0.1    // N/A
CAM_TYPE        0
CAM_EDGES       -200 -100 150 80
CAM_DRIFT       1.25   // per-frame
```

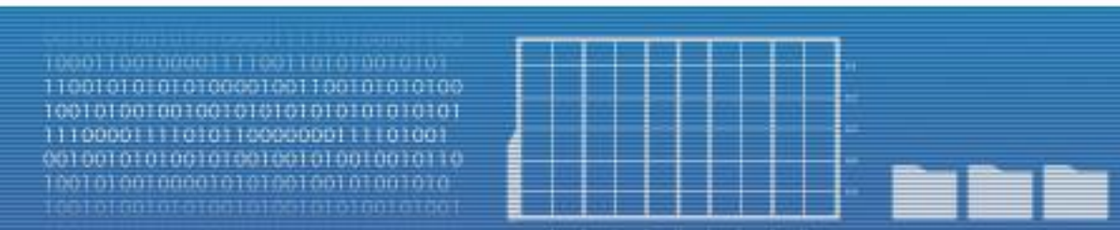
DISCS

Homework

CAM_TYPE = camera type to implement
    0 - position-lock
    1 - position-lock with edge-snapping
    2 - camera-window
    3 - camera-window with position-snapping
    4 - camera-window with platform-snapping

CAM_EDGES = upper-left corner x and y, and lower-right corner x and y
    in the example, upper-left corner is (-200, -100)
    while lower-right corner is (150, 80)

CAM_DRIFT = rate at which camera will align with the player character

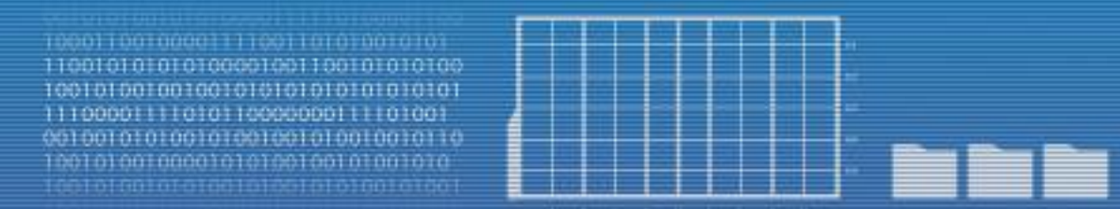


Homework

Note that CAM_EDGES will only be used for CAM_TYPEs 1, 2, 3, and 4.

For CAM_TYPE 1, CAM_EDGES defines a rectangular area of the game world that the camera's center cannot go outside of.

For CAM_TYPEs 2, 3, and 4, CAM_EDGES defines the rectangular area relative to the camera's current center. You are REQUIRED to draw this rectangular area's borders.

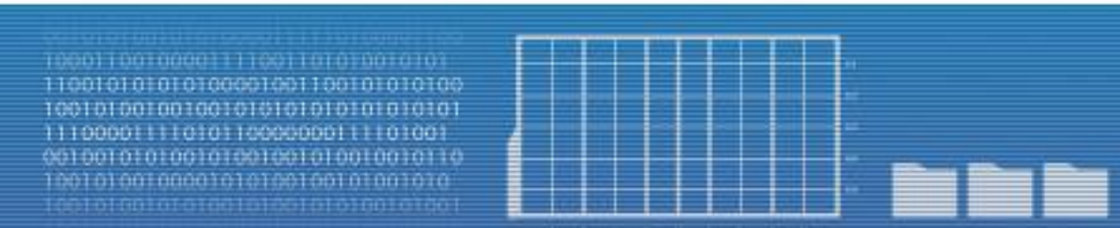


Homework

Also note that CAM_DRIFT will only be used for CAM_TYPEs 3 and 4.

For CAM_TYPE 3, CAM_DRIFT is the speed for each axis at which the camera's center moves towards the player if the player is not pushing the camera-window edges for that specific axis.

For CAM_TYPE 4, CAM_DRIFT only applies to the y-axis and only if the player is grounded.



References

- http://www.gamasutra.com/blogs/ItayKeren/20150511/243083/Scroll_Back_The_Theory_and_Practice_of_Cameras_in_SideScrollers.php