CPSC 427
Video Game Programming

IO and the Observer Pattern

Helge Rhodin

Announcements and outlook

Guest lecturers (all broadcasted to the classroom):
- ***Oct 20 - SkyBox (Rendering)***
- ***Oct 27 - Nvidia (Raytracing, incl. Neural Networks)***
- ***Nov 17 - Charm Games (VR games)***

Face-to-face grading A1: register! (if selected at ‘random’)
- ***https://docs.google.com/spreadsheets/d/1CMZsU9mmMIj36xcbobDn5N1AOUHDTX9BO14BtYw4hio/edit?usp=sharing***

Next deadline: M1

Be prepared for team presentations (like the oral pitch) and M1 face-to-face grading

Assignment A2 posted

Feature clarifications

- ***New Organization feature (worth 10 points, take is serious)***
- ***Particle effects (basic)***
  - Create particle locations and their motion on the CPU (smoke, fire, dirt…)
  - Render one Quad at every particle location
  - Create a shader (similar to light-up of the salmon that renders the particle in local object coordinates; can also be a texture)
  - ~~glDrawArraysInstanced~~ (old technique, no longer used)
- ***Advanced particle effects (counts as a new feature)***
  - *Use the OpenGL point rendering function instead of quads*

Feature clarification

- ***~~Geometry~~ shader -> Vertex shader (mandatory)***
  - *Move vertices around, e.g., deform on collision?*

- Geometry shader (advanced, optional)
  - *Create new vertices, e.g., subdivision, explosion, …*

Today

Recap: collisions

Communication between systems:

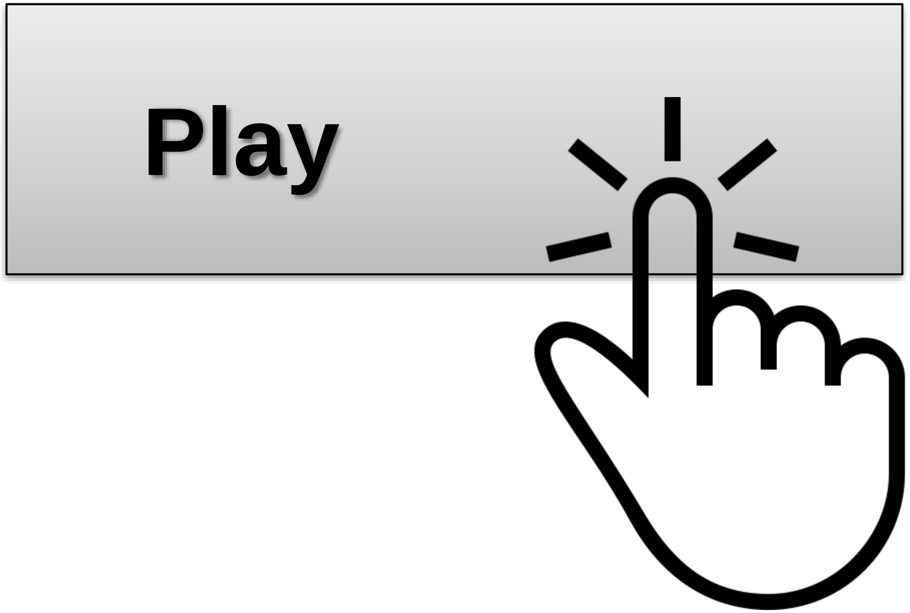
- ***The observer pattern***

If time permits, some more UI and UX

Recap: Collisions

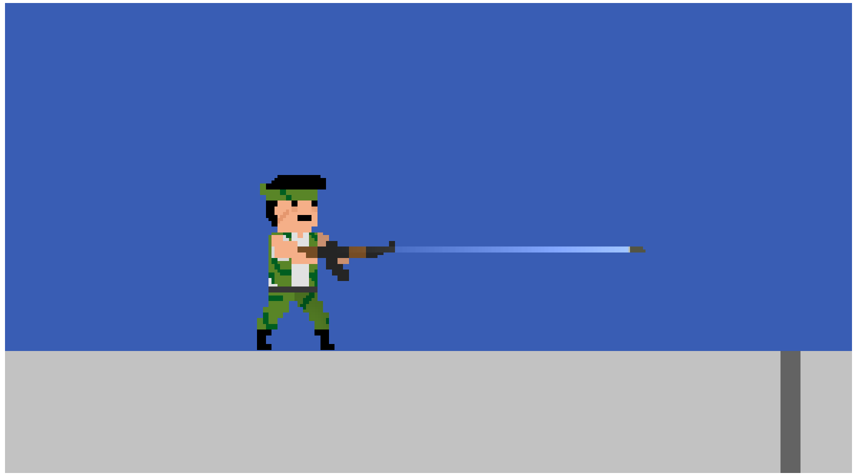
Motivation: Object selection

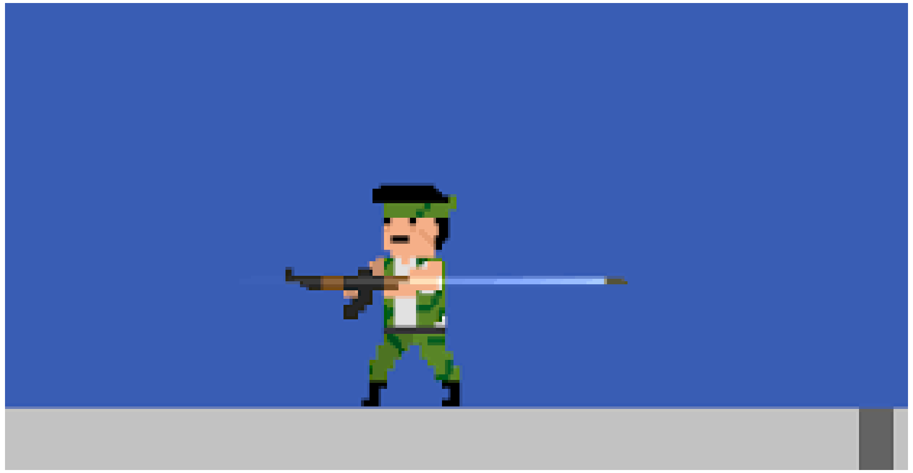
- ***Point inside object boundary?***



Motivation: Bullet trajectories

- ***Line-object or point-object intersection?***



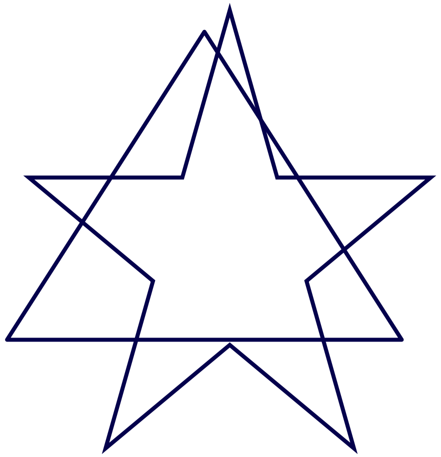


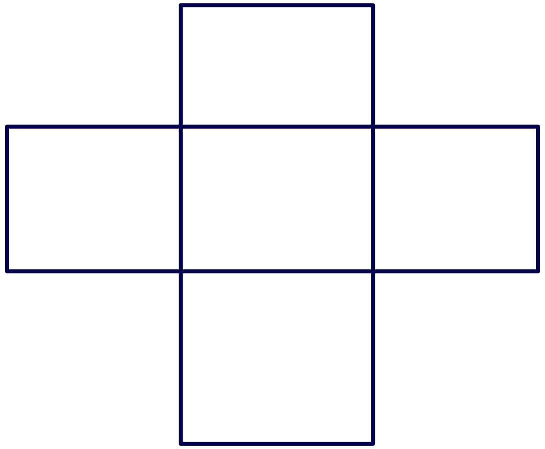
https://forum.unity.com/threads/2d-platformer-shooting.365971/

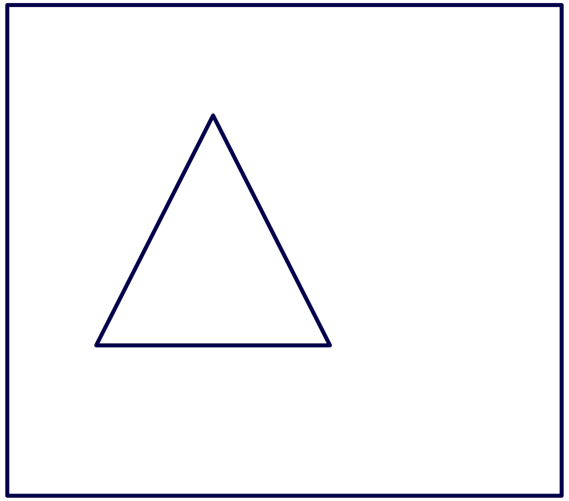
Collision Configurations?

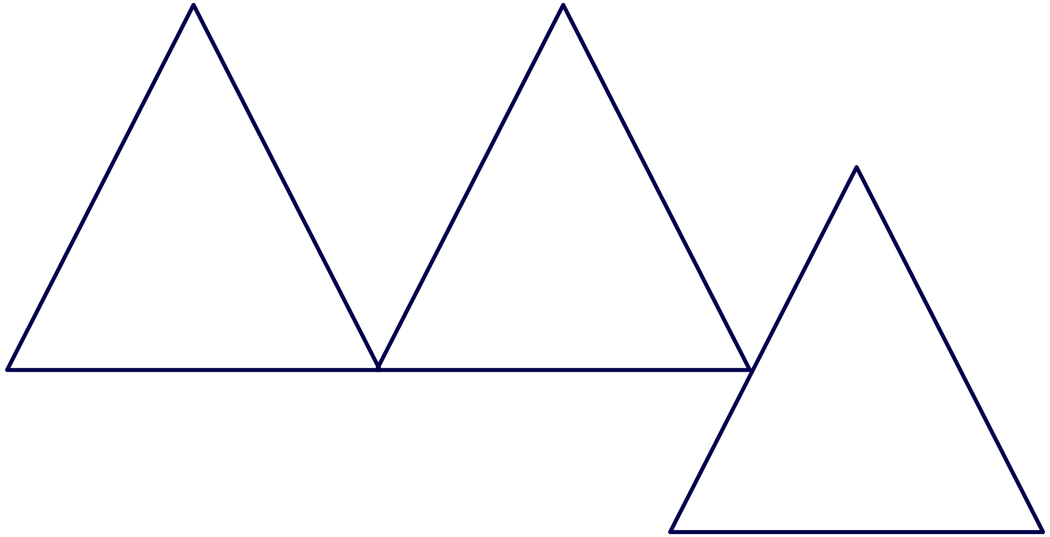
- Segment/Segment Intersection
  - *Point **on** Segment*

- Polygon inside polygon



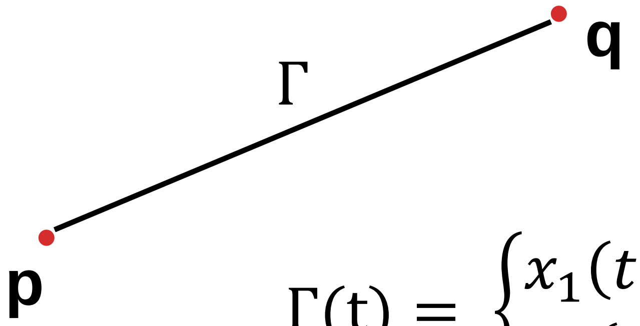






Lines & Segments

Segment Γ from $\mathbf{p} = (x_0, y_0)$ ***to*** $\mathbf{q} = (x_1, y_1)$



$$\Gamma(t) = \begin{cases} x_1(t) = x_0 + (x_1 - x_0)t \\ y_1(t) = y_0 + (y_1 - y_0)t \end{cases} \quad t \in [0,1]$$

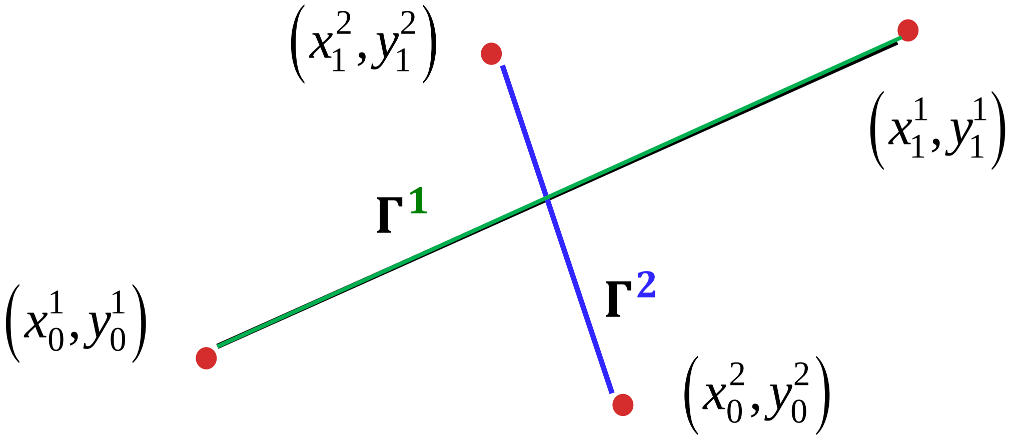
Find the line through $\boldsymbol{p} = (\boldsymbol{x_0}, \boldsymbol{y_0})$ ***and*** $\boldsymbol{q} = (\boldsymbol{x_1}, \boldsymbol{y_1})$***?***

- Parametric: $\Gamma(t), t \in (-\infty, \infty)$
- Implicit: $Ax + By + C = 0$
  - *Solve 2 equations in 2 unknowns (substitute* $A^2+B^2=1$*)*

Line-Line Intersection

$$\Gamma^1 = \begin{cases} x^1(t) = x_0^1 + (x_1^1 - x_0^1)t \\ y^1(t) = y_0^1 + (y_1^1 - y_0^1)t \end{cases} \quad t \in [0,1]$$

$$\Gamma^2 = \begin{cases} x^2(r) = x_0^2 + (x_1^2 - x_0^2)r \\ y^2(r) = y_0^2 + (y_1^2 - y_0^2)r \end{cases} \quad r \in [0,1]$$



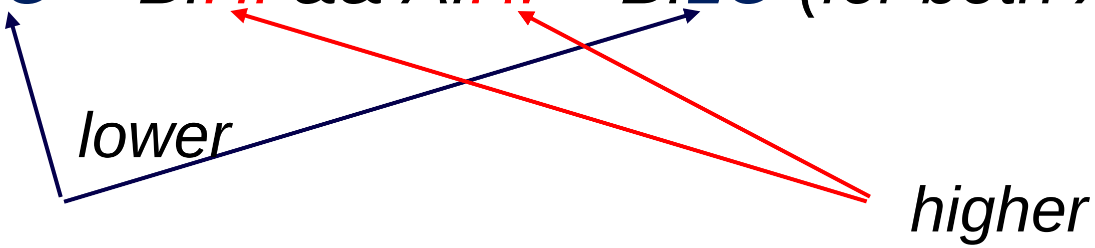
Intersection: *x* & *y* values equal in both representations - two linear equations in two unknowns (*r*,*t*)

$$\boxed{\begin{aligned} x_0^1 + (x_1^1 - x_0^1)t &= x_0^2 + (x_1^2 - x_0^2)r \\ y_0^1 + (y_1^1 - y_0^1)t &= y_0^2 + (y_1^2 - y_0^2)r \end{aligned}}$$

Question: What is the meaning if the solution gives *r*,*t* < 0 or *r*,*t* >1 ?

Bounding Volume Intersection

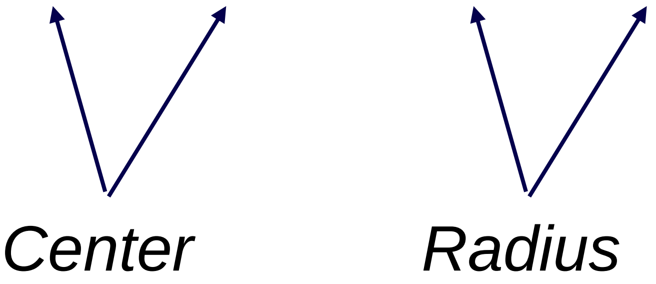
- Axis aligned bounding box (AABB)
  - *A.LO<=B.HI && A.HI>=B.LO (for both X and Y)*



lower

higher

- Circles
  - $\|A.C - B.C\| < A.R + B.R$



Center

Radius

Different intersection types

Point – line (which side of line)

Point – object (contained)

Line – line

Line – object

Object – object

Moving objects

- Sweep – test intersections against before/after segment
    - *Avoid "jumping through" objects*
    - *How to do efficiently?*
- Boxes?
- Spheres?

Getting stuck – Collision resolution

- ***Solutions:***
- If collision is detected, stop at position of previous step
  - *Can still cause locked situations since objects can go back in time*

- Only count collision when objects are moving towards each other
  - *Requires to consider motion direction of both objects*
    - What if one is still?

Complexity?

What is the complexity of checking collision between all objects in your game (naïve version)?

A: Linear in number of objects

B: Quadratic

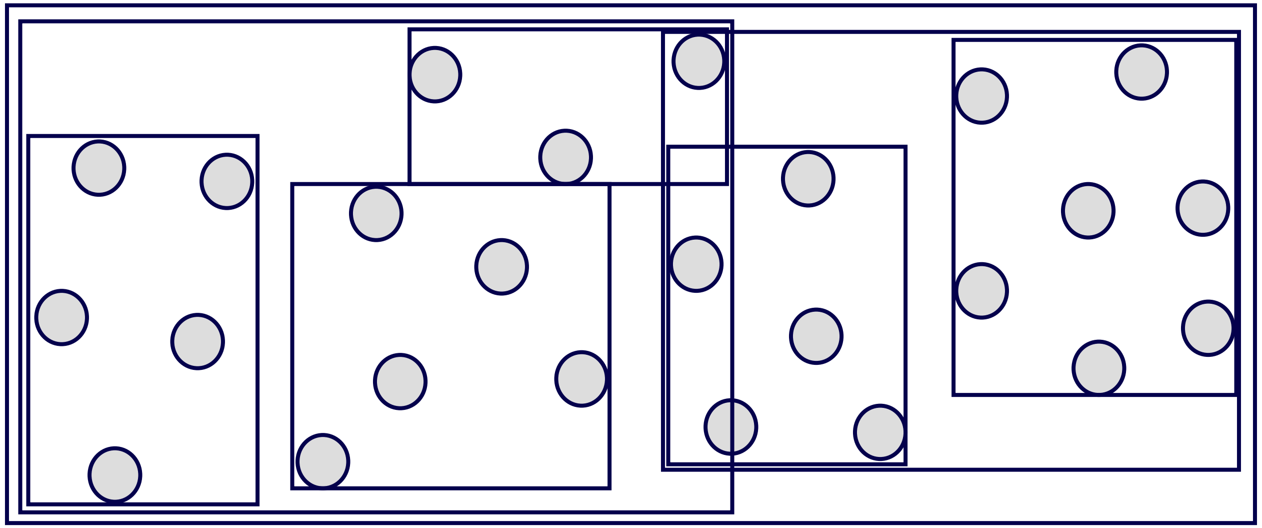
C: Logarithmic

D: Exponential

Hierarchical Bounding Volumes

Bound Bounding Volumes:

- Use (hierarchical) bounding volumes for groups of objects

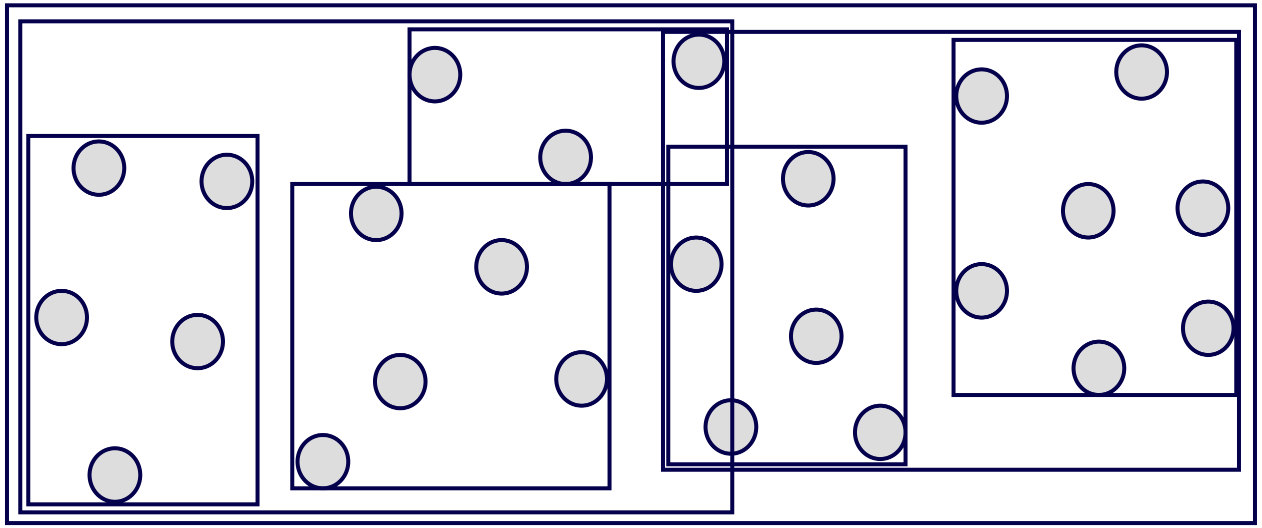


- How to group boxes?
    - *Closest*
    - *Most jointly compact (how?)*

Hierarchical Bounding Volumes

Bound Bounding Volumes:

- Use (hierarchical) bounding volumes for groups of objects



- Challenge: dynamic data…
  - *Need to update hierarchy efficiently*

Spatial Subdivision DATA STRUCTURES

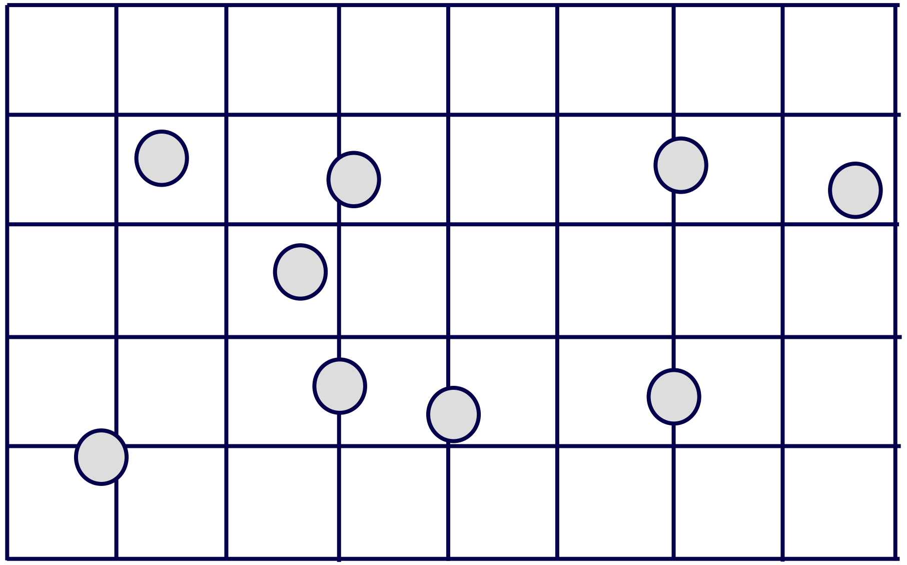
- Subdivide space (bounding box of the “world”)
- Hierarchical
  - *Subdivide each sub-space (or only non-empty sub-spaces)*
- Lots of methods
  - *Grid, Octree, k-D tree, (BSP tree)*

© Alla Sheffer, Helge Rhodin

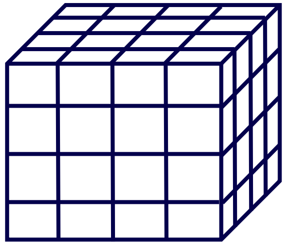
Regular Grid

Subdivide space into rectangular grid:

- Associate every object with the cell(s) that it overlaps with
- Test collisions only if cells overlap



In 3D: regular grid of cubes (voxels):

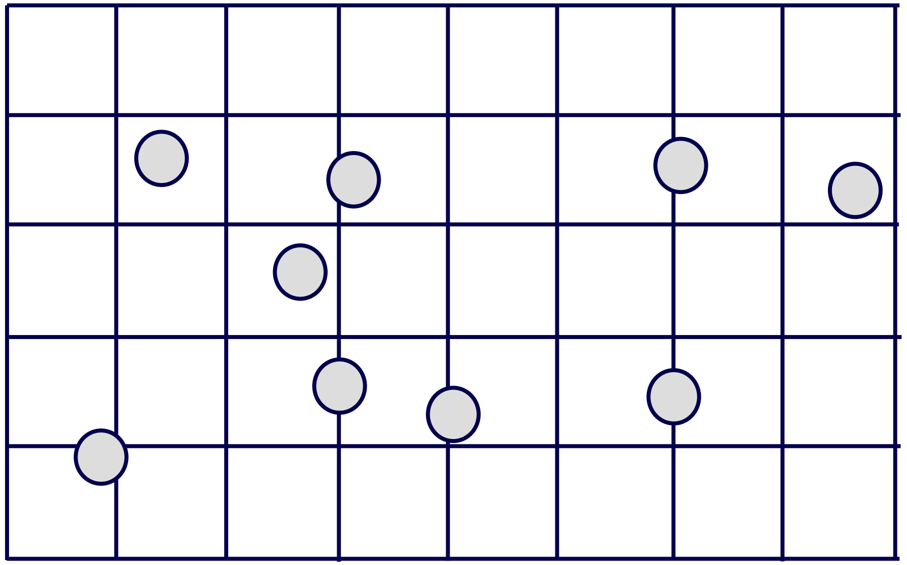


© Alla Sheffer, Helge Rhodin

Creating a Regular Grid

Steps:

- Find bounding box of scene
- Choose grid resolution in x, y, z
- Insert objects
- Objects that overlap multiple cells get referenced by all cells they overlap



Regular Grid Discussion

Advantages?

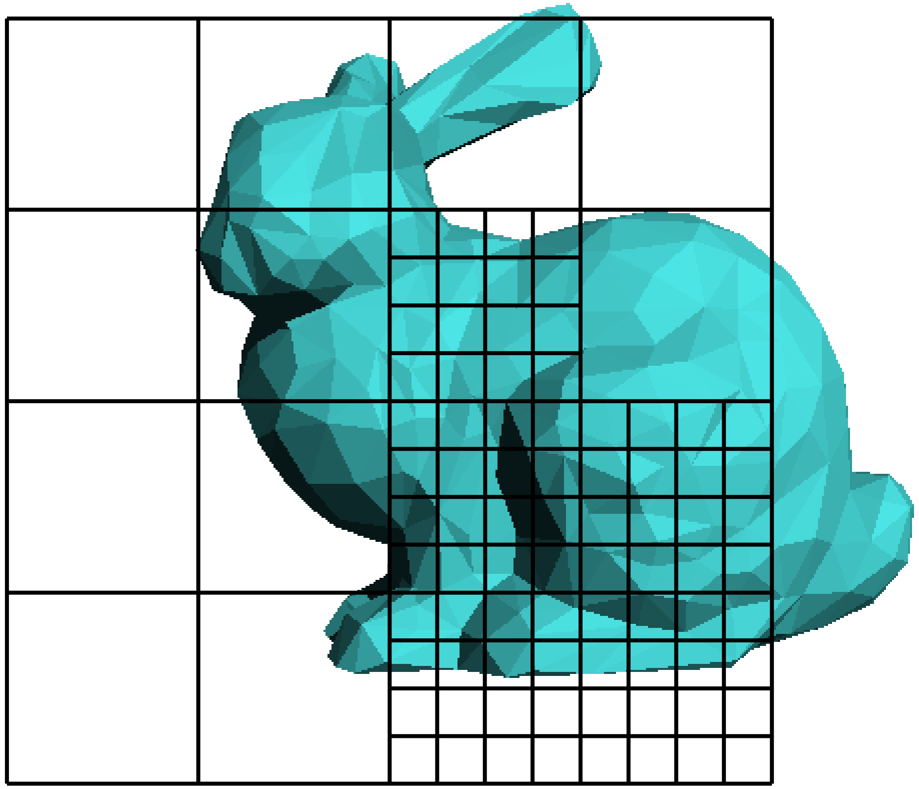
- Easy to construct
- Easy to traverse

Disadvantages?

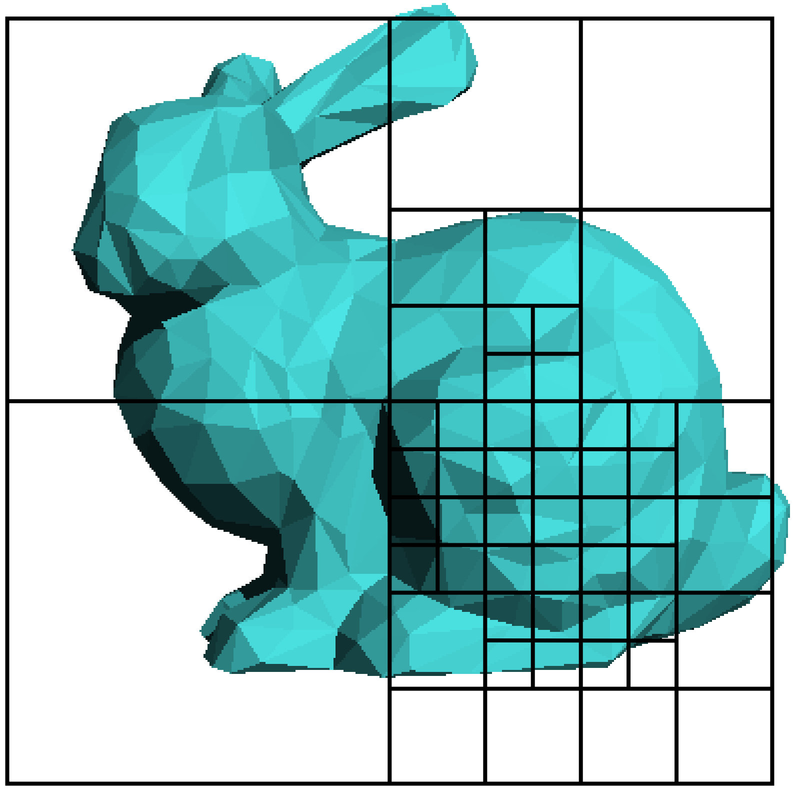
- May be only sparsely filled
- Geometry may still be clumped

Adaptive Grids

- **Subdivide until each cell contains no more than *n* elements, or maximum depth *d* is reached**



Nested Grids



Octree/(Quadtree)

- This slide is curtsey of Fredo Durand at MIT

Collision Resolution

Today: simplified example

Upcoming lecture:

Physics-based simulation

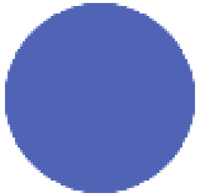
Basic Particle Simulation (first try)

How to compute the change in velocity?

$$d_t = t_{i+1} - t_i$$

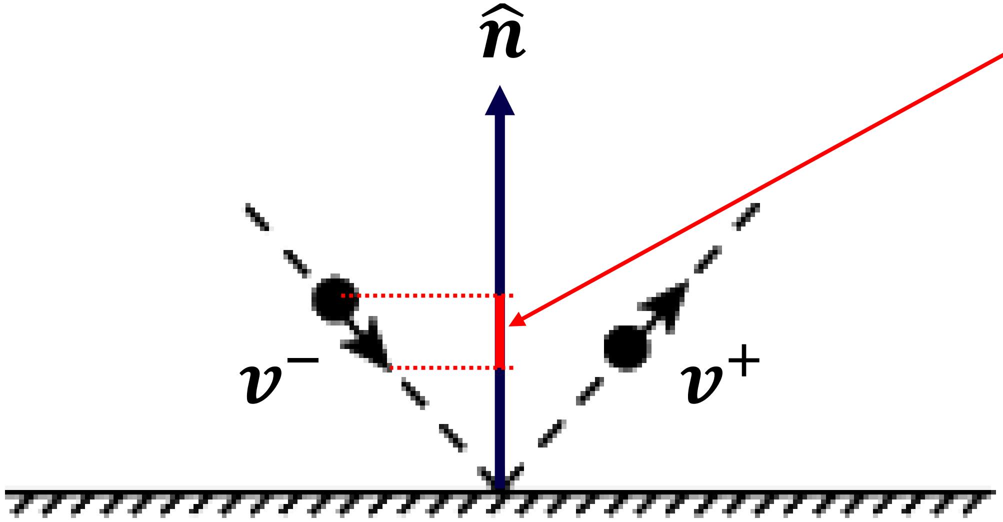
$$\vec{\boldsymbol{v}}_{i+1} = \vec{\boldsymbol{v}}_i + \Delta v$$

$$\vec{\boldsymbol{p}}_{i+1} = \vec{\boldsymbol{p}}(t_i) + \vec{\boldsymbol{v}}_i d_t$$



Particle-Plane Collisions

- ***Change in direction of normal***



Velocity along normal (v projected on normal by the dot product)

Frictionless

Apply change along normal (magnitude times direction)

$$\Delta v = 2\overline{(v^- \circ \hat{n})}\,\overline{\hat{n}}$$

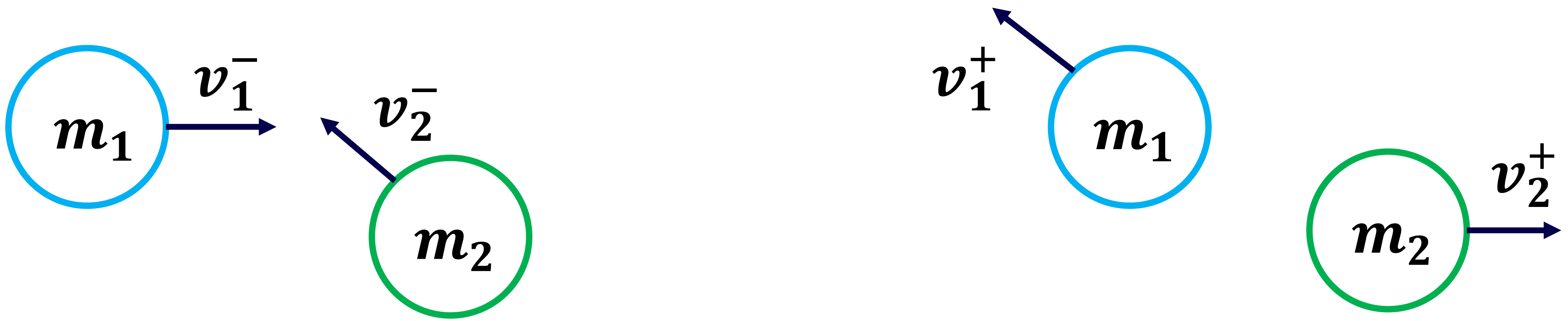
$$v^+ = v^- + \Delta v$$

Loss of energy

$$\Delta v = (1 + \epsilon)(v^- \circ \hat{n})\hat{n}$$

Particle-Particle Collisions (spherical objects)

Before collision **After**



Response:

$$v_1^+ = v_1^- - \frac{2m_2}{m_1+m_2}\frac{\langle v_1^- - v_2^-\rangle\cdot\langle p_1 - p_2\rangle}{\|p_1 - p_2\|^2}\langle p_1 - p_2\rangle$$

$$v_2^+ = v_2^- - \frac{2m_1}{m_1+m_2}\frac{\langle v_2^- - v_1^-\rangle\cdot\langle p_2 - p_1\rangle}{\|p_2 - p_1\|^2}\langle p_2 - p_1\rangle$$

- This is in terms of velocity
- Upcoming lectures:
  derivation via impulse and forces

CPSC 427
Video Game Programming

IO and the Observer Pattern

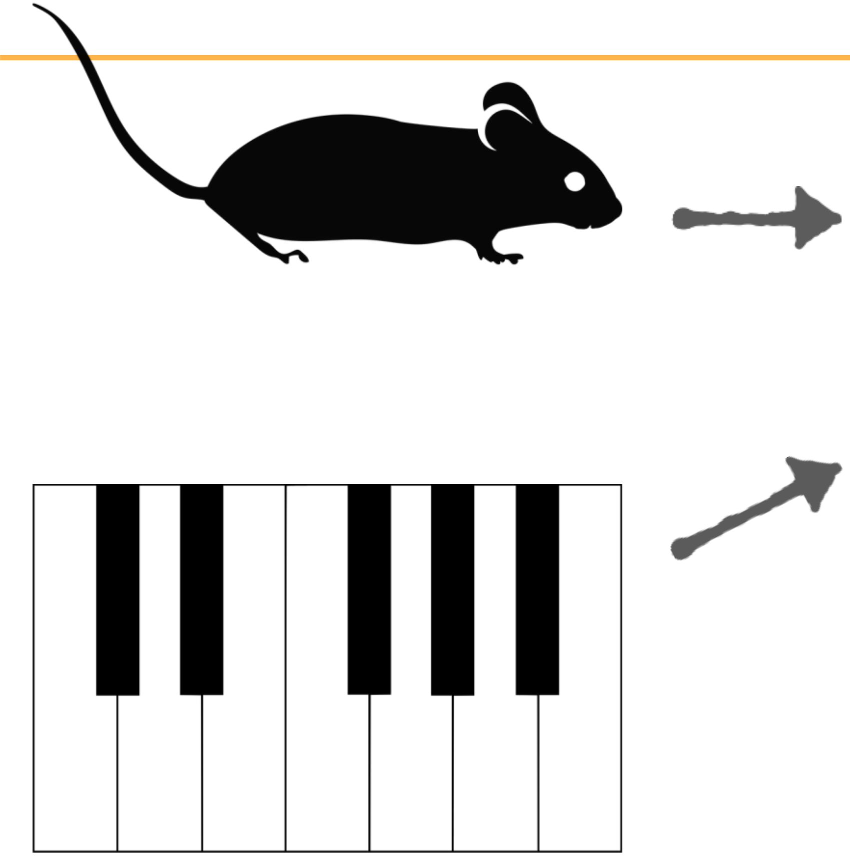
Helge Rhodin

Mainloop

```
int main(int argc, char* argv[]) {
…
2. Mainloop:
while (!world.is_over()) {
   2.1 Event processing
   2.2 Game state update
   2.3 Rendering a frame
}
…
}
```

© Alla Sheffer, Helge Rhodin

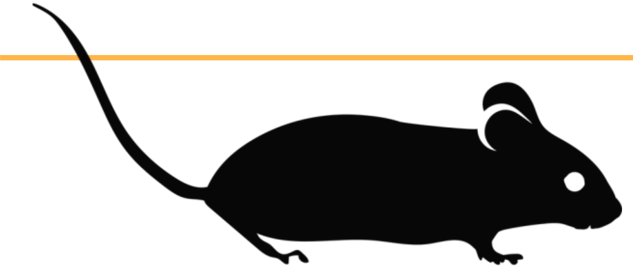
Event Processing

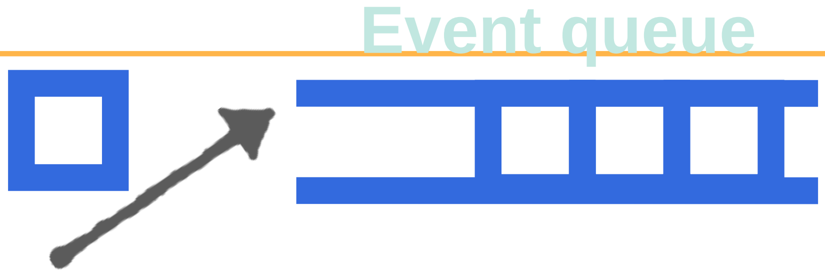


Mouse event,
Keyboard event,
etc.

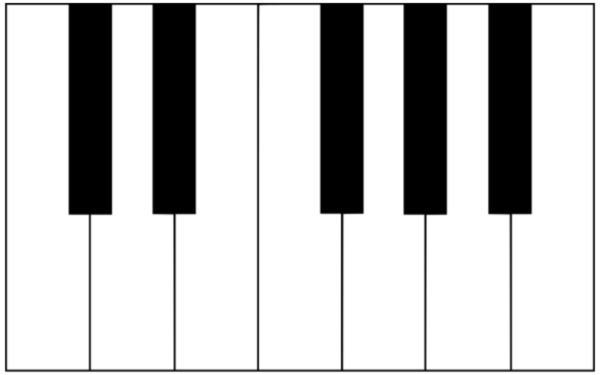
Credits:
https://pixabay.com/en/mouse-mouse-silhouette-lab-mouse-2814846/
https://svgsilh.com/image/25711.html

Event Processing: Event Queuing



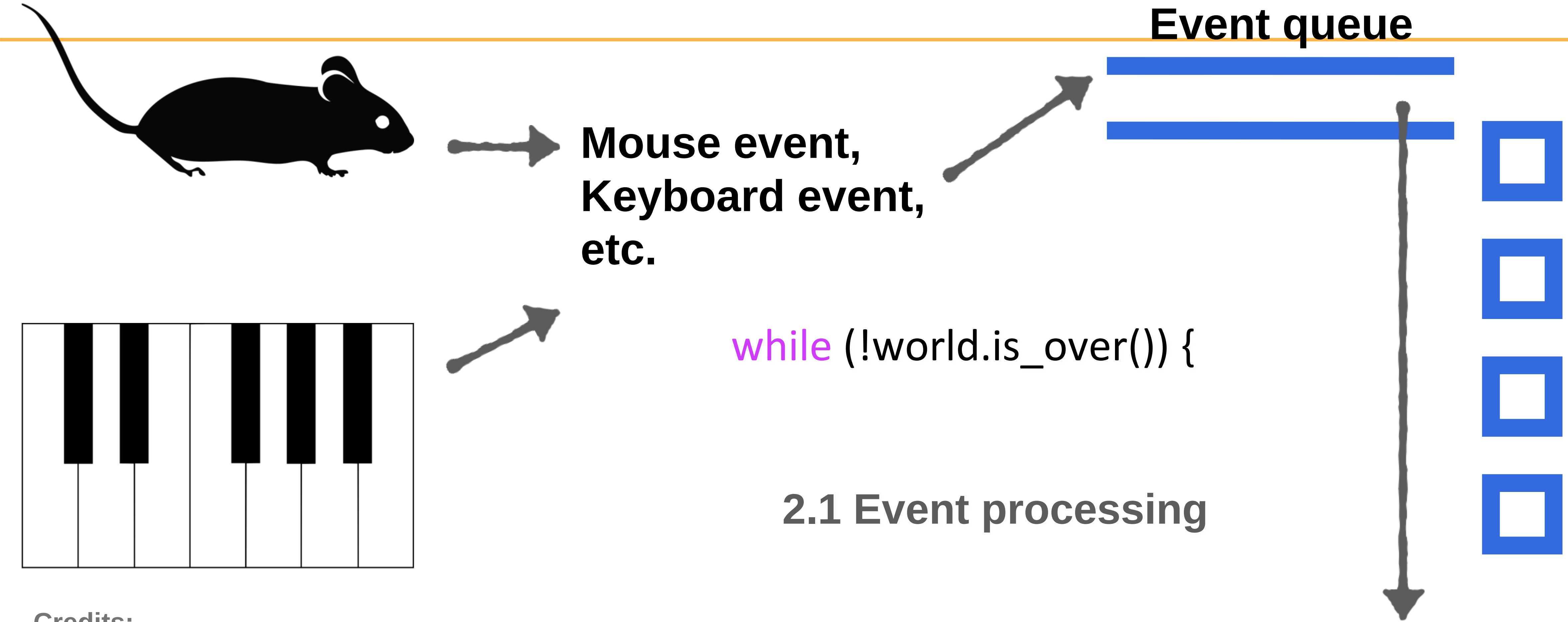


Mouse event,
Keyboard event,
etc.



Credits:
https://pixabay.com/en/mouse-mouse-silhouette-lab-mouse-2814846/
https://svgsilh.com/image/25711.html

Event Processing: Event Polling



Event queue

Mouse event,
Keyboard event,
etc.

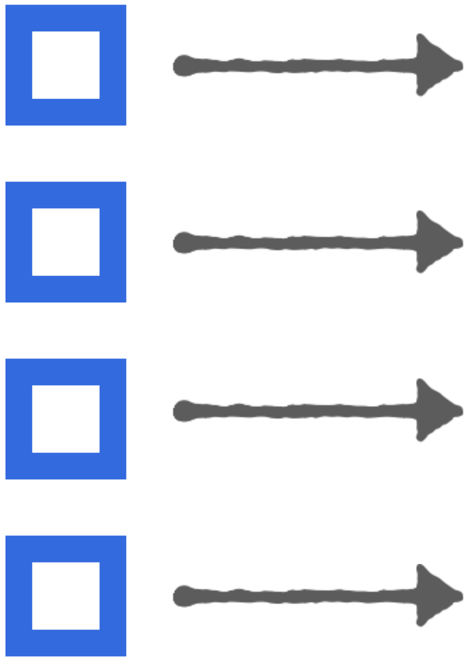
```
while (!world.is_over()) {
```

2.1 Event processing

```
glfwPollEvents();
```

Credits:
https://pixabay.com/en/mouse-mouse-silhouette-lab-mouse-2814846/
https://svgsilh.com/image/25711.html

Event Processing: Event Callback



GLFW calls corresponding callbacks:

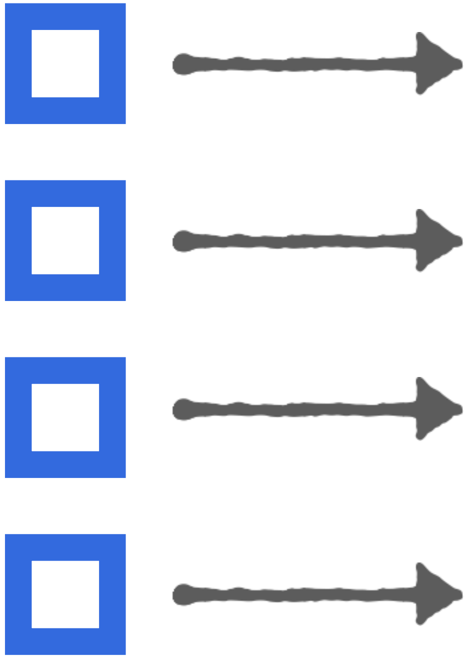
- void World::on_key(GLFWwindow*, int key, int, int action, int mod)

  —-> You set salmon motion here.

- void World::on_mouse_move(GLFWwindow* window, double xpos, double ypos)

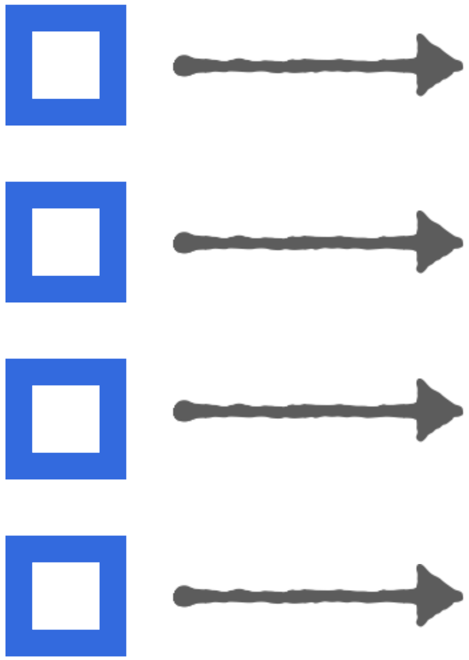
  —-> You set salmon rotation.

Event Processing: Event Callback



How does GLFW know which callback to call?

Event Processing: Event Callback



How does GLFW know which callback to call?

—-> Registered in initialization:

`world.init(…)`

```
glfwSetKeyCallback
glfwSetCursorPosCallback
```

Mainloop

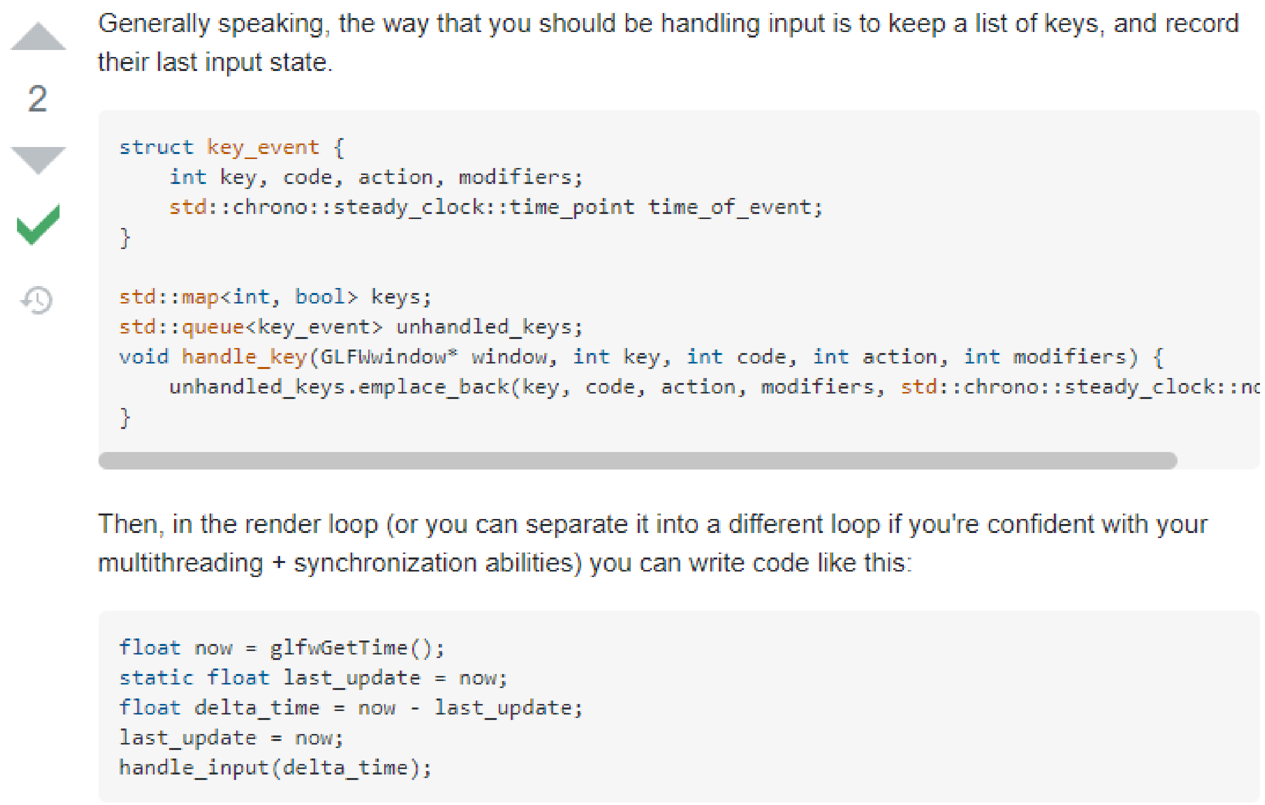
```
int main(int argc, char* argv[]) {
…
2. Mainloop:
while (!world.is_over()) {
   2.1 Event processing
   2.2 Game state update
   2.3 Rendering a frame
}
…
}
```

`glfwPollEvents`

- ***Asynchronous?***
- ***Reference:*** ***https://www.glfw.org/docs/3.0/group__window.html#ga37bd57223967b4211d60ca1a0bf3c832***
  - "This function processes only those events that have already been received and then returns immediately. Processing events will cause the window and input callbacks associated with those events to be called."
    - synchronous!
- "On some platforms, certain callbacks may be called outside of a call to one of the event processing functions."
  - asynchronous! :/

Workaround?

- ***https://stackoverflow.com/questions/36579771/glfw-key-callback-synchronization***



2

Generally speaking, the way that you should be handling input is to keep a list of keys, and record their last input state.

```
struct key_event {
    int key, code, action, modifiers;
    std::chrono::steady_clock::time_point time_of_event;
}

std::map<int, bool> keys;
std::queue<key_event> unhandled_keys;
void handle_key(GLFWwindow* window, int key, int code, int action, int modifiers) {
    unhandled_keys.emplace_back(key, code, action, modifiers, std::chrono::steady_clock::no
}
```

Then, in the render loop (or you can separate it into a different loop if you're confident with your multithreading + synchronization abilities) you can write code like this:

```
float now = glfwGetTime();
static float last_update = now;
float delta_time = now - last_update;
last_update = now;
handle_input(delta_time);
```

The Observer Pattern

- ***Gang of Four (GoF)***
  - *Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides*
  - ***Design Patterns: Elements of Reusable Object-Oriented Software*** (1994)

- ***A pattern described by the GoF***
- event-driven
  - *clients register for an event*

Good ref (object oriented):
https://gameprogrammingpatterns.com/observer.html

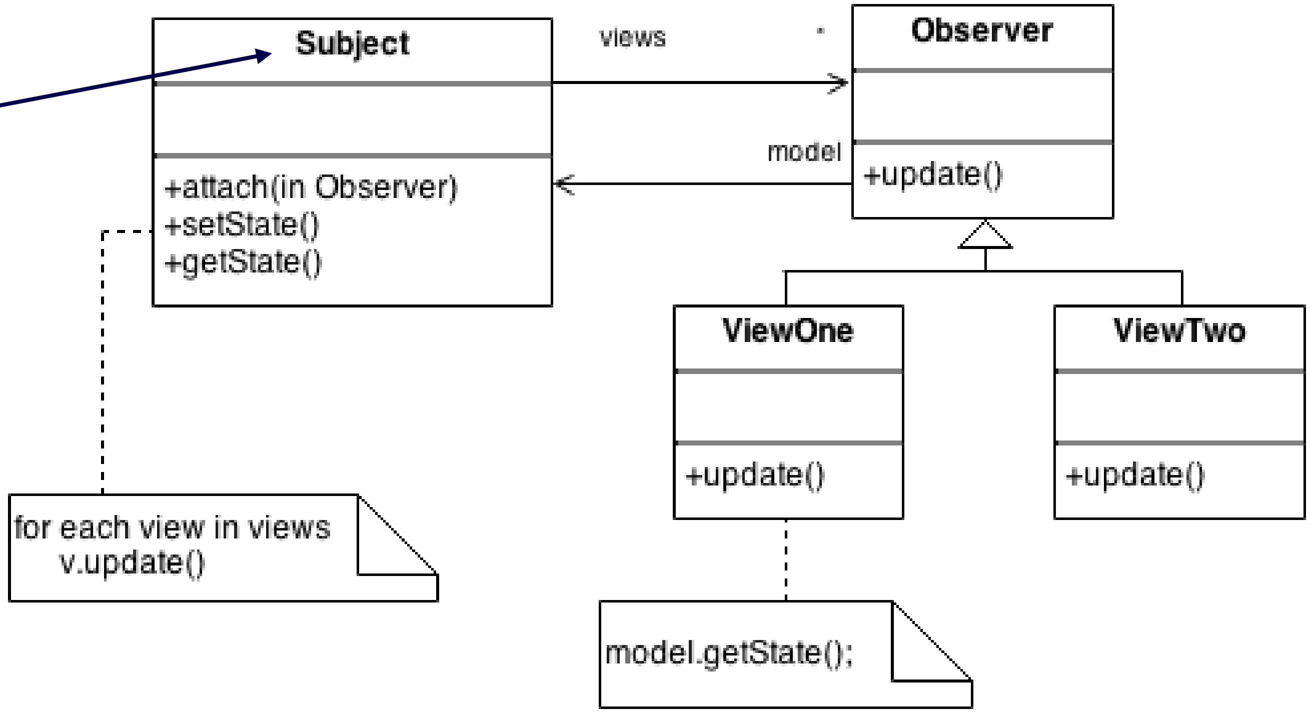
Use Cases

- ***Rewards***
- ***Communication between systems (in ECS)***
- ***User input***
- ***???***

Observer Pattern – OOP

- ***Define a common interface***
- ***All observers inherit from that interface***

Called Subject by GoF



Lambda Functions

Definition:

- auto y = [] (int first, int second) { return first + second; };

Call: int z = y(1+3);

- Infers return type for simple functions (single return statement)
    - otherwise

```
auto y = [] (int first, int second) -> int { return first + second; };
```

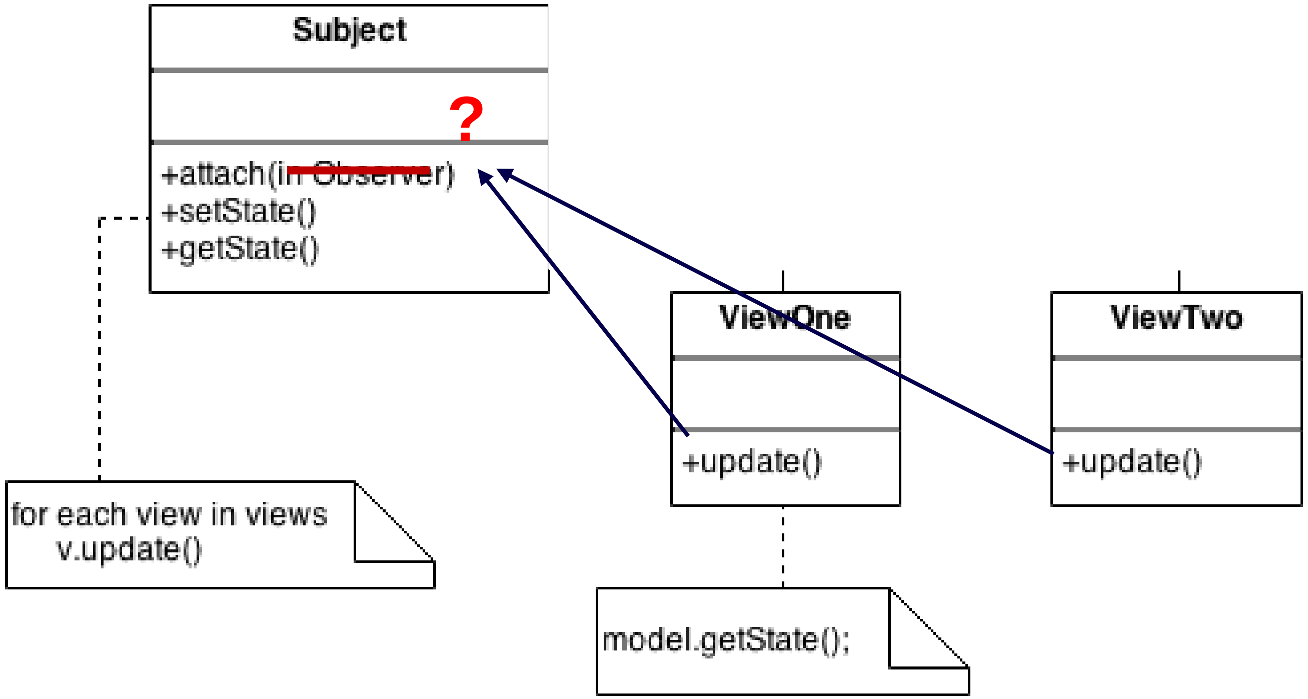
- Can capture variables from the surrounding scope.

```
int scale;
auto y = [] (int first, int second) -> int { return scale*first + second; };

auto y = [&] (int first, int second) -> int { return scale*first + second; };
```

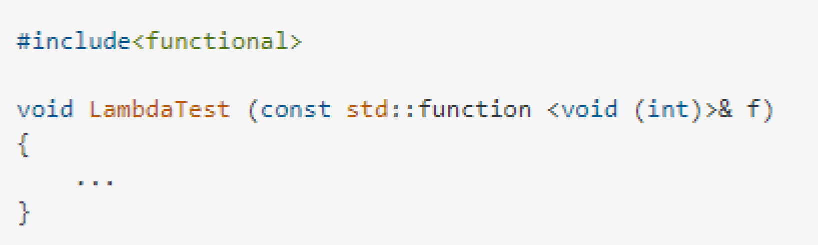
Observer Pattern – With Functions

- ***function with matching signature instead of class***



A function that accepts a function

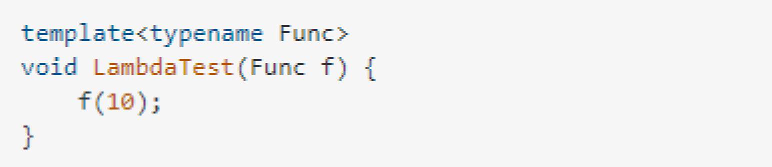
- ***Using std::function***



```
#include<functional>

void LambdaTest (const std::function <void (int)>& f)
{
    ...
}
```

- ***Using templates***

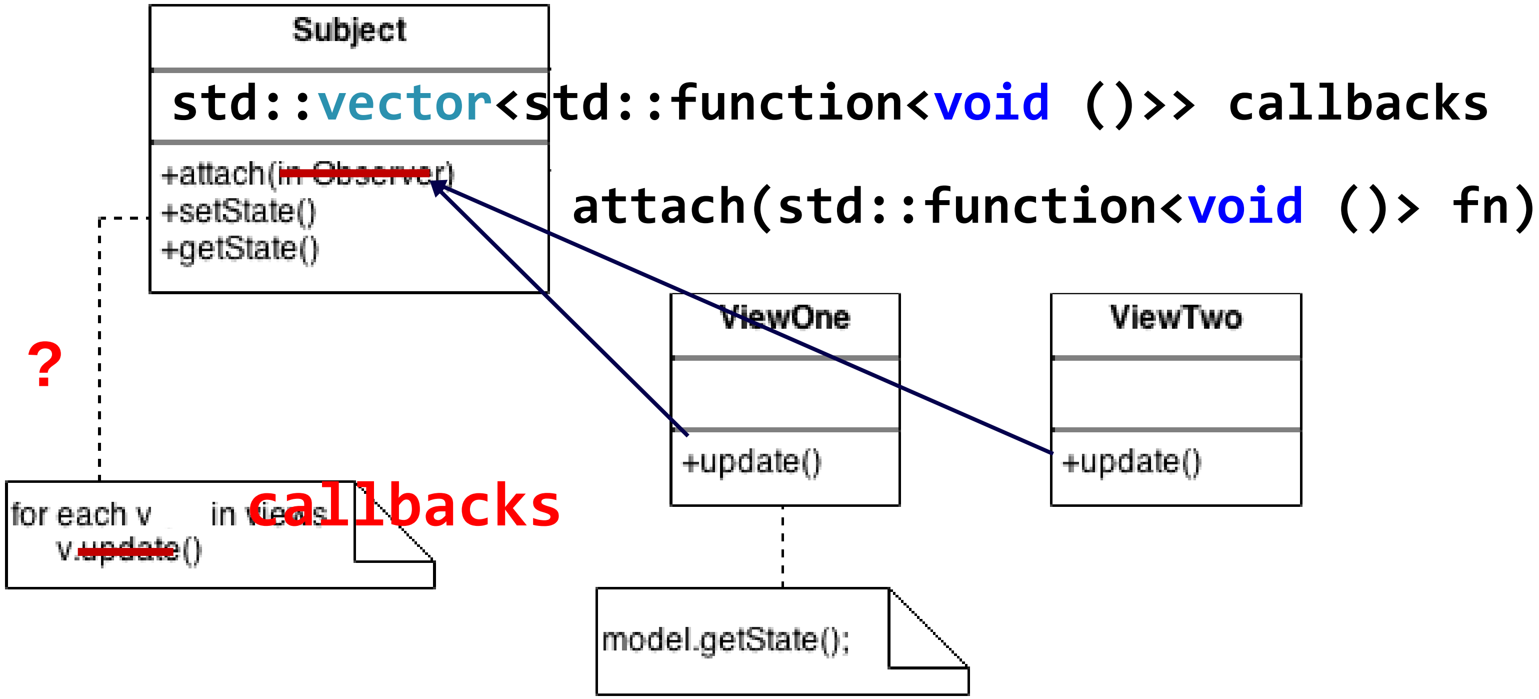


```
template<typename Func>
void LambdaTest(Func f) {
    f(10);
}
```

  - *use templates to accept any argument with an operator()*

Observer Pattern – With Functions

- ***function with matching signature instead of class***

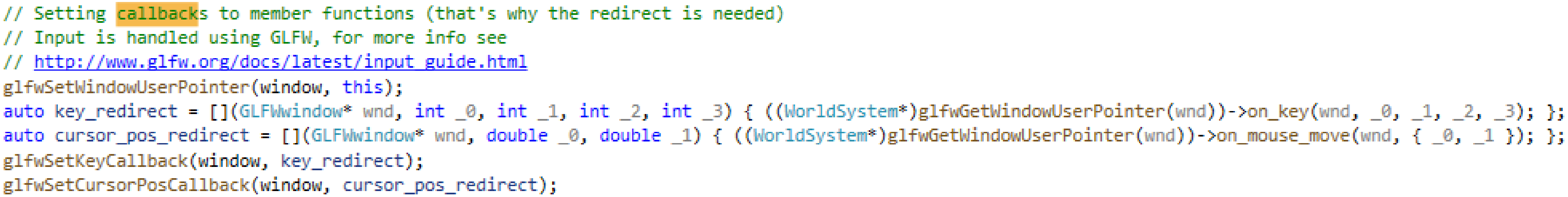


```
std::vector<std::function<void ()>> callbacks
```

```
attach(std::function<void ()> fn)
```

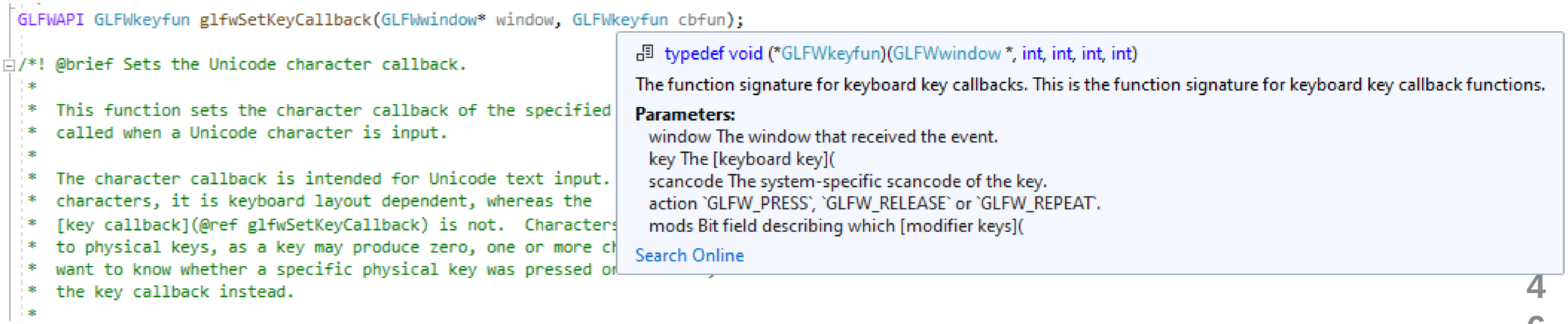
IO in our Template

- *????????*



```
// Setting callbacks to member functions (that's why the redirect is needed)
// Input is handled using GLFW, for more info see
// http://www.glfw.org/docs/latest/input_guide.html
glfwSetWindowUserPointer(window, this);
auto key_redirect = [](GLFWwindow* wnd, int _0, int _1, int _2, int _3) { ((WorldSystem*)glfwGetWindowUserPointer(wnd))->on_key(wnd, _0, _1, _2, _3); };
auto cursor_pos_redirect = [](GLFWwindow* wnd, double _0, double _1) { ((WorldSystem*)glfwGetWindowUserPointer(wnd))->on_mouse_move(wnd, { _0, _1 }); };
glfwSetKeyCallback(window, key_redirect);
glfwSetCursorPosCallback(window, cursor_pos_redirect);
```

- *Function signature*



```
GLFWAPI GLFWkeyfun glfwSetKeyCallback(GLFWwindow* window, GLFWkeyfun cbfun);

/*! @brief Sets the Unicode character callback.
 *
 *  This function sets the character callback of the specified
 *  called when a Unicode character is input.
 *
 *  The character callback is intended for Unicode text input.
 *  characters, it is keyboard layout dependent, whereas the
 *  [key callback](@ref glfwSetKeyCallback) is not.  Characters
 *  to physical keys, as a key may produce zero, one or more ch
 *  want to know whether a specific physical key was pressed or
 *  the key callback instead.
 *
```

typedef void (*GLFWkeyfun)(GLFWwindow *, int, int, int, int)

The function signature for keyboard key callbacks. This is the function signature for keyboard key callback functions.

Parameters:

window The window that received the event.
key The [keyboard key](
scancode The system-specific scancode of the key.
action `GLFW_PRESS`, `GLFW_RELEASE` or `GLFW_REPEAT`.
mods Bit field describing which [modifier keys](

Search Online

Performance?

- ***Isn’t this slow?***