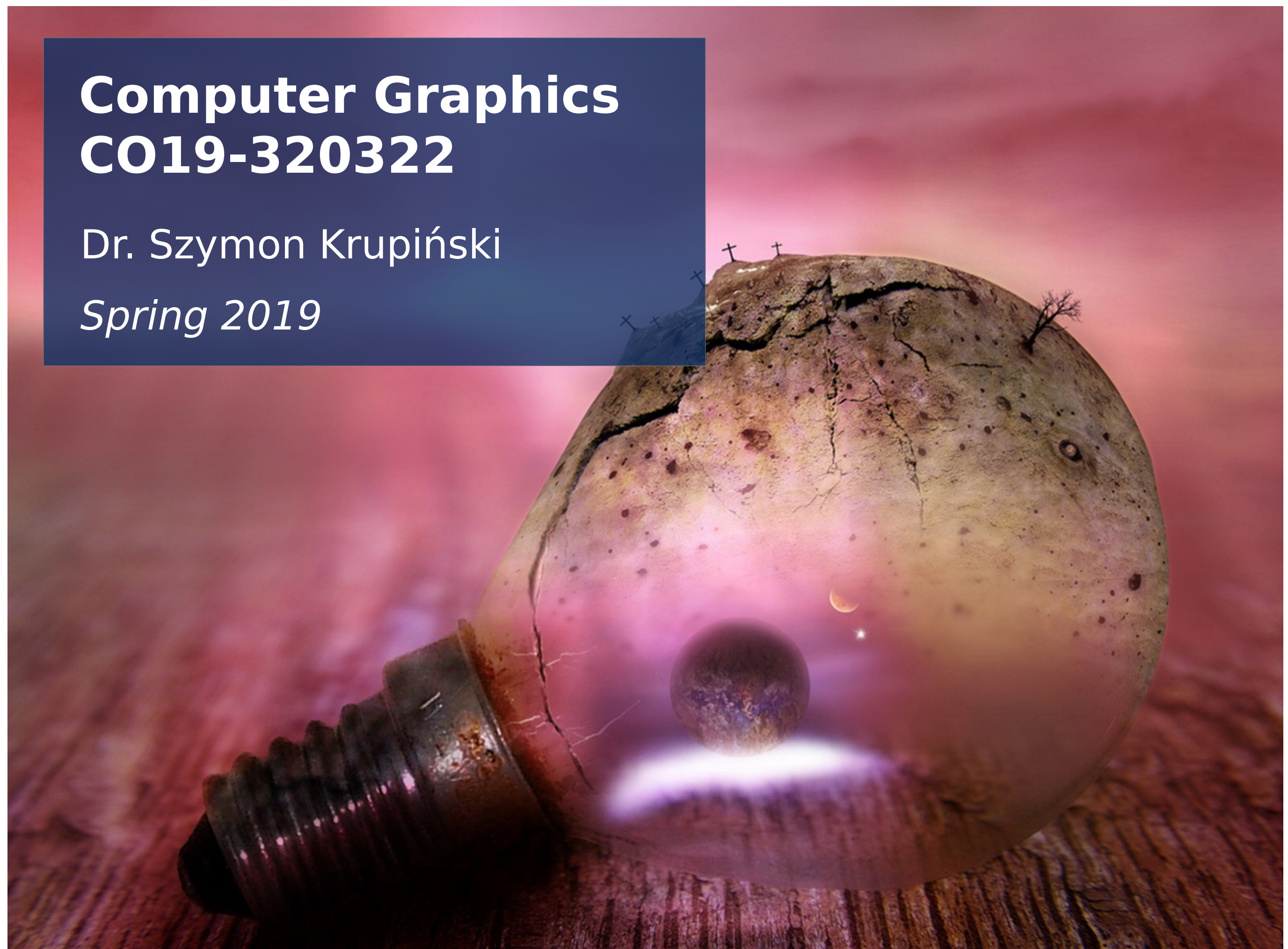
Lecture 4: Rasterisation 1



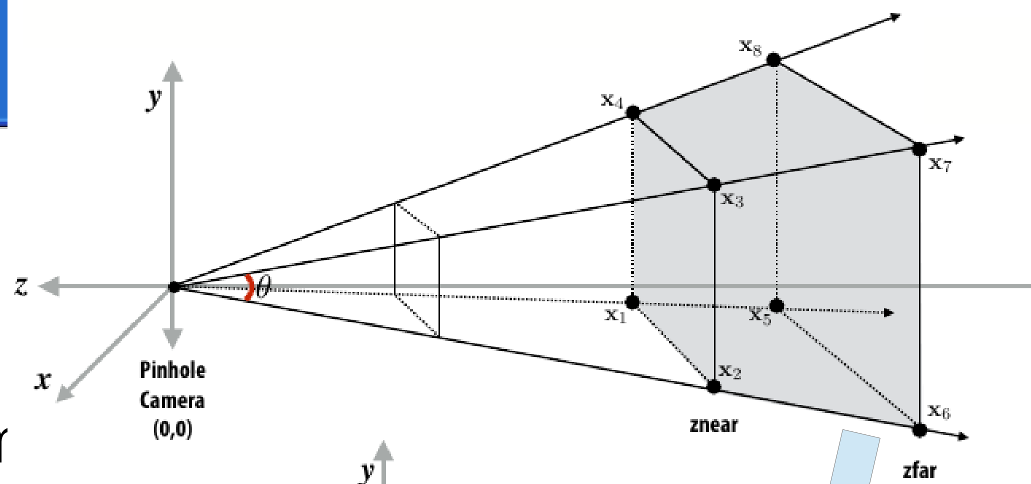
Computer Graphics
CO19-320322

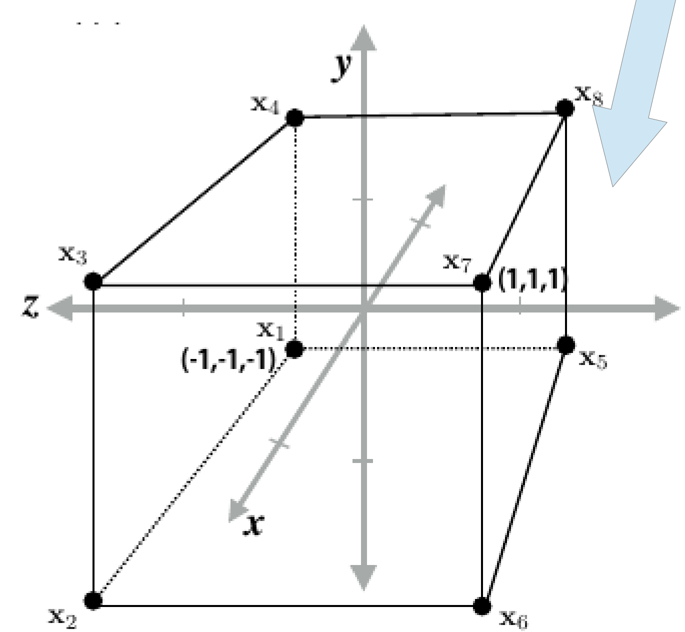
Dr. Szymon Krupiński

Spring 2019

From yesterday: camera and projection

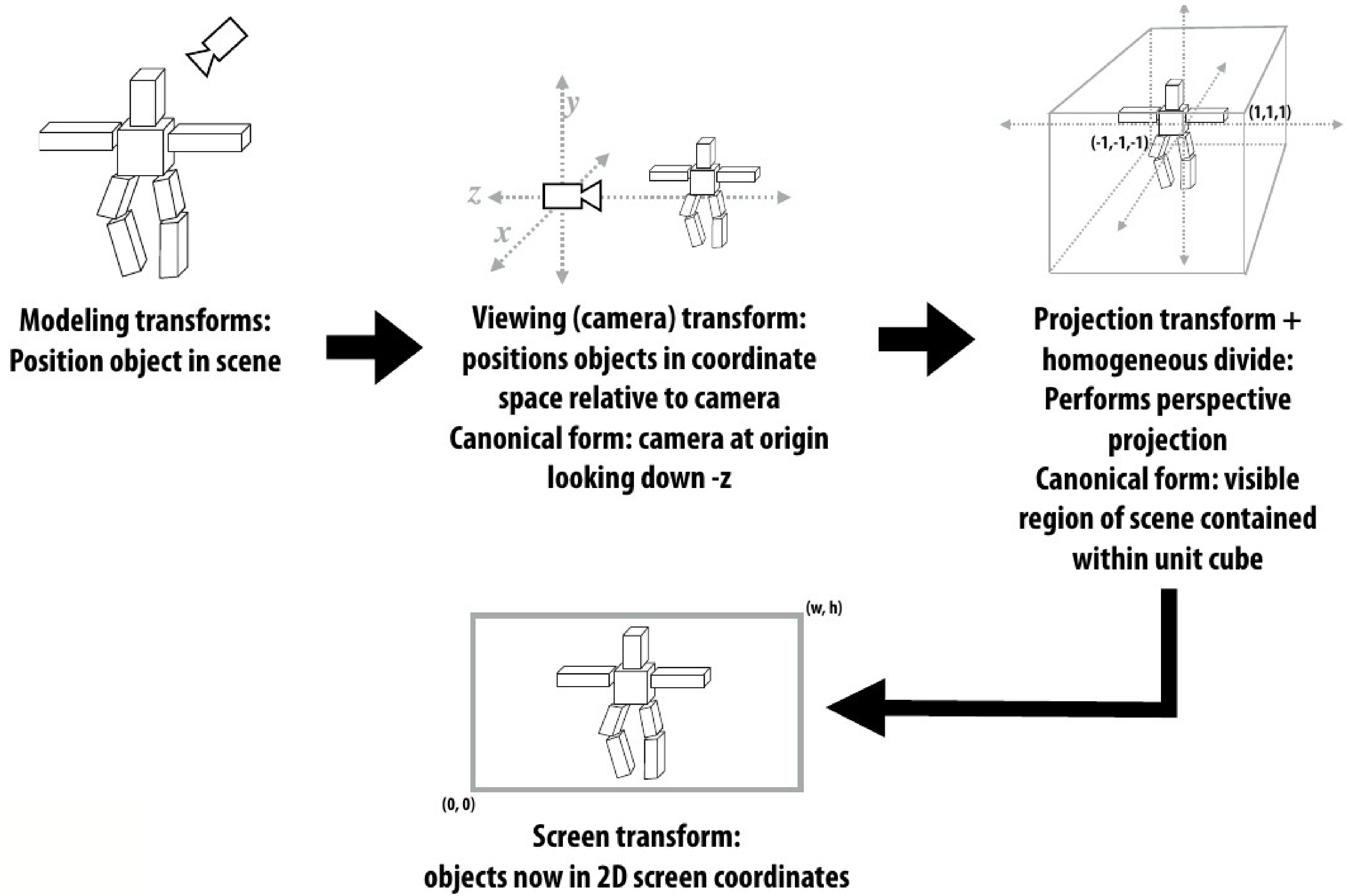
- A camera can be modelled using the pinhole model
- It has a number of important parameters which are often visualised as a camera frustrum defining the geometry of the projection
- We constructed a mathematical expression for perspective transform which yields uniform coordinates. Good news, it's a matrix multiplication + homogeneous coordinates normalisation → can be applied efficiently
- To be tested: what happens if we apply the projection to the frustrum itself?





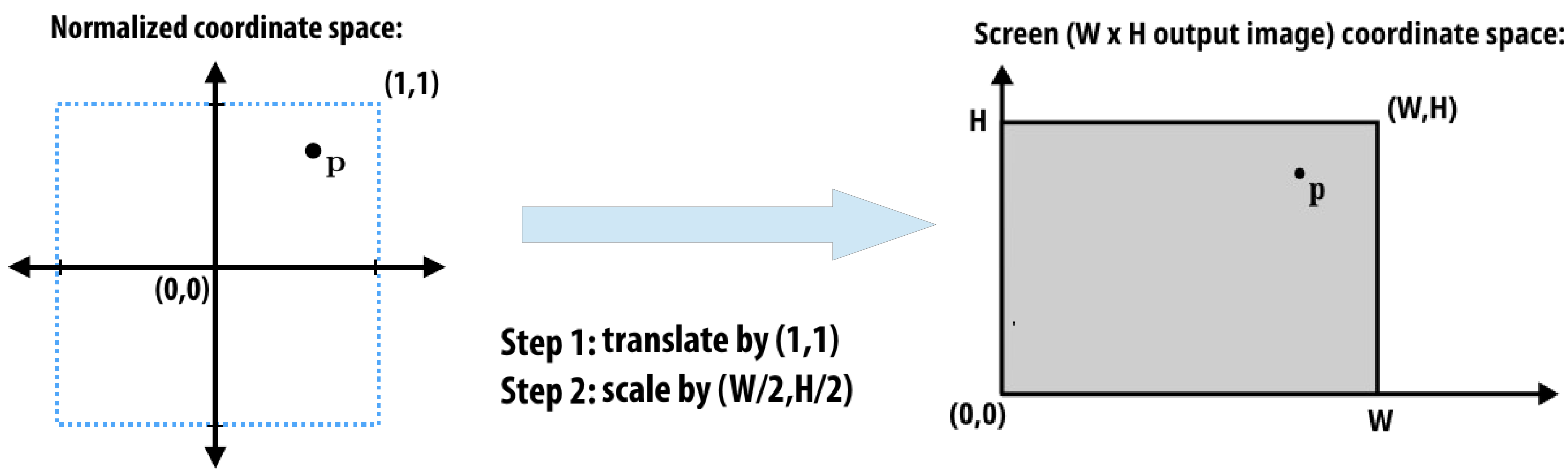
The unit box is where all the (visible) action happens!

Where are we now?



Screen transformation

- We didn’t talk about the details of the last transformation because… it’s rather simple! (but watch out for screen coordinate convention which differs from library to library):

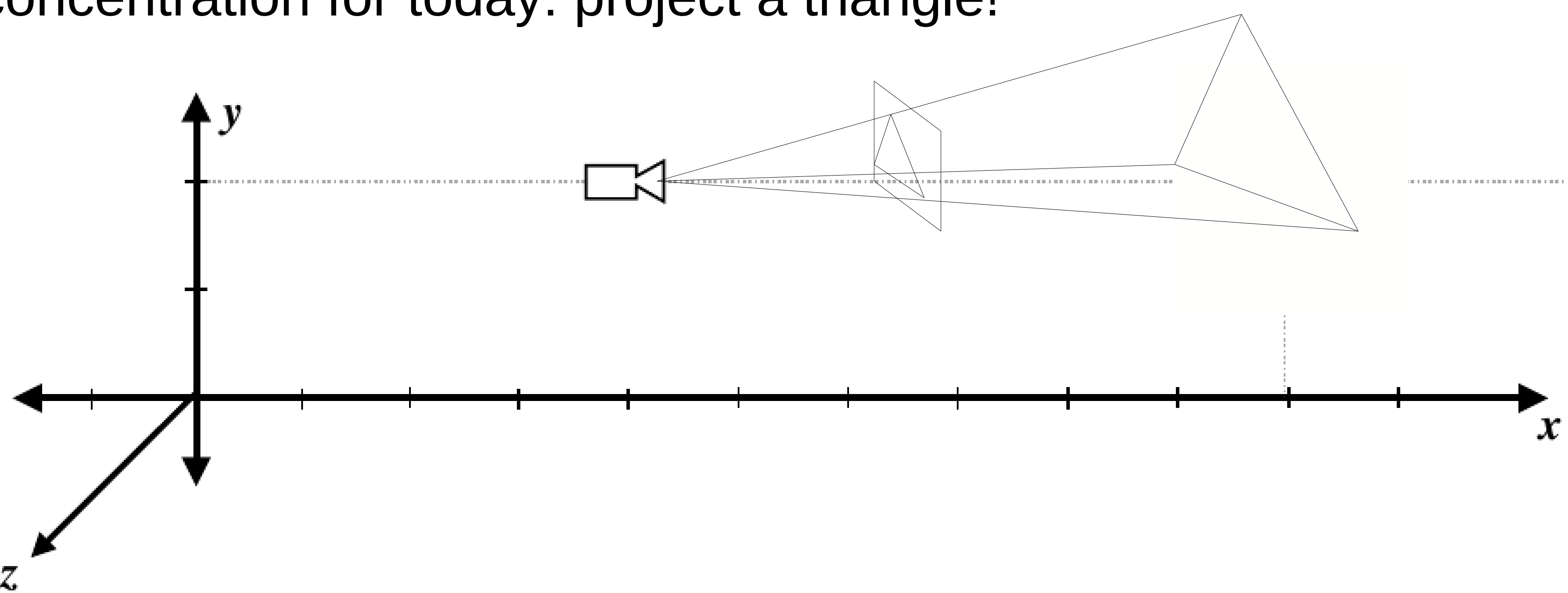


This comes out of the normalised projection

These are the screen/image pixel coordinates of point p

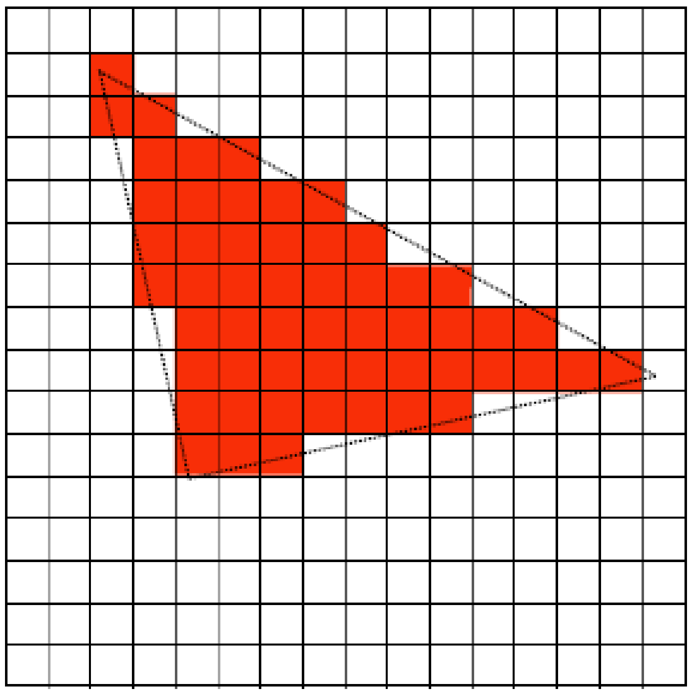
Facing the triangles

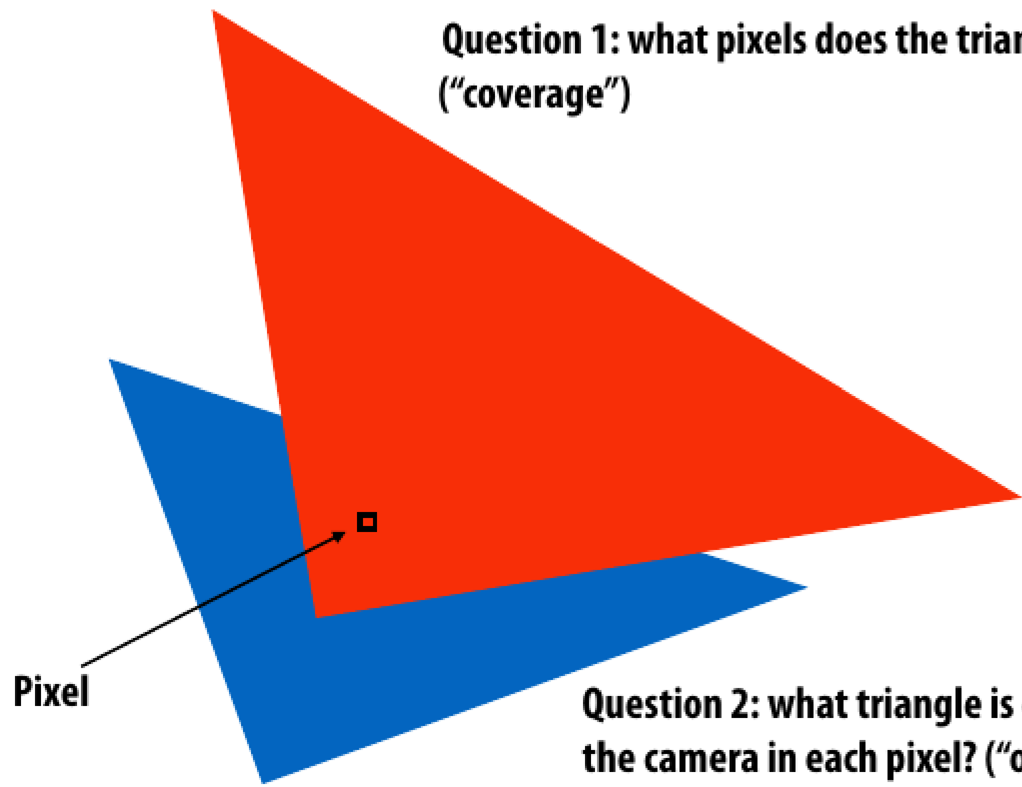
- Up to now, we were transforming points, imagining wire meshes built upon them
- We had a first glance at a non-trivial problem: how to determining which pixels should be attributed to a given part of geometry
- Our concentration for today: project a triangle!



Facing the triangles

Question 3: how does one draw it realistically?





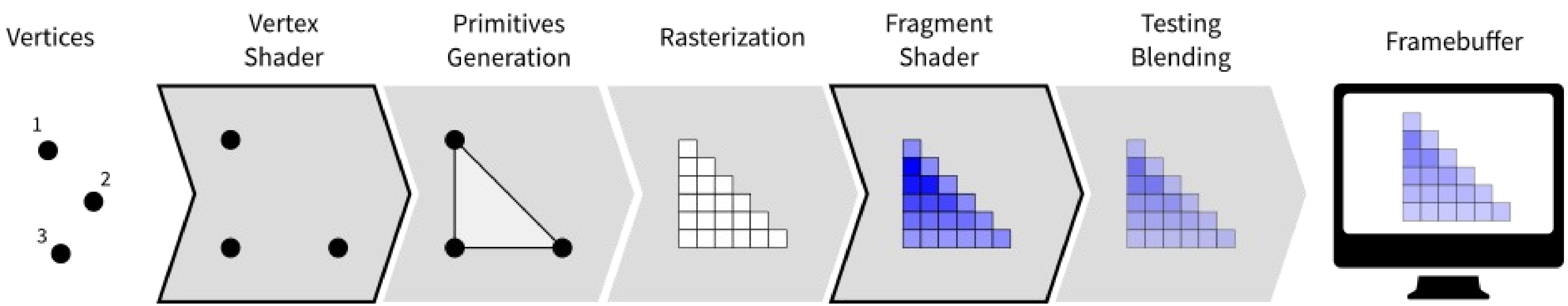
Question 1: what pixels does the triangle overlap? ("coverage")

Question 2: what triangle is closest to the camera in each pixel? ("occlusion")

Question 4: how to cover all geometry and attribute all pixels on the screen?

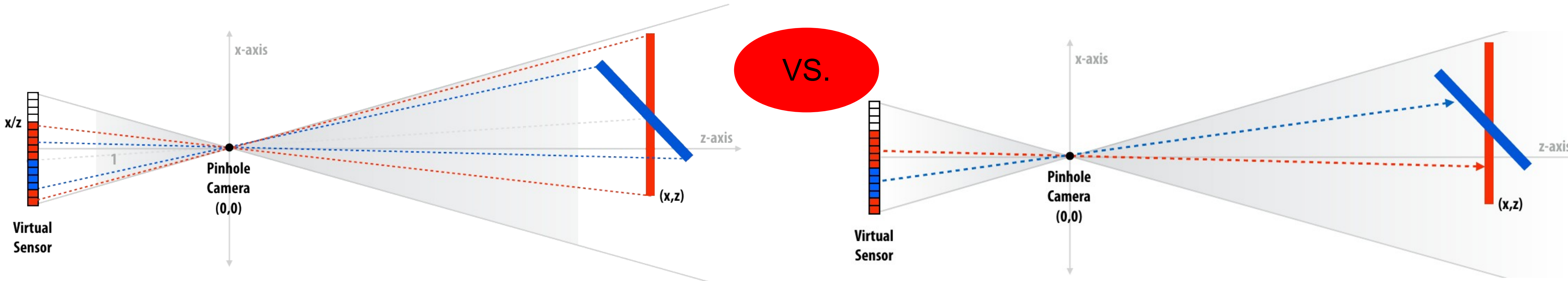
Yet another broader look

- Without much ado, we are beginning to cover some steps of the modern CG pipeline, such as implemented in the recent versions of OpenGL



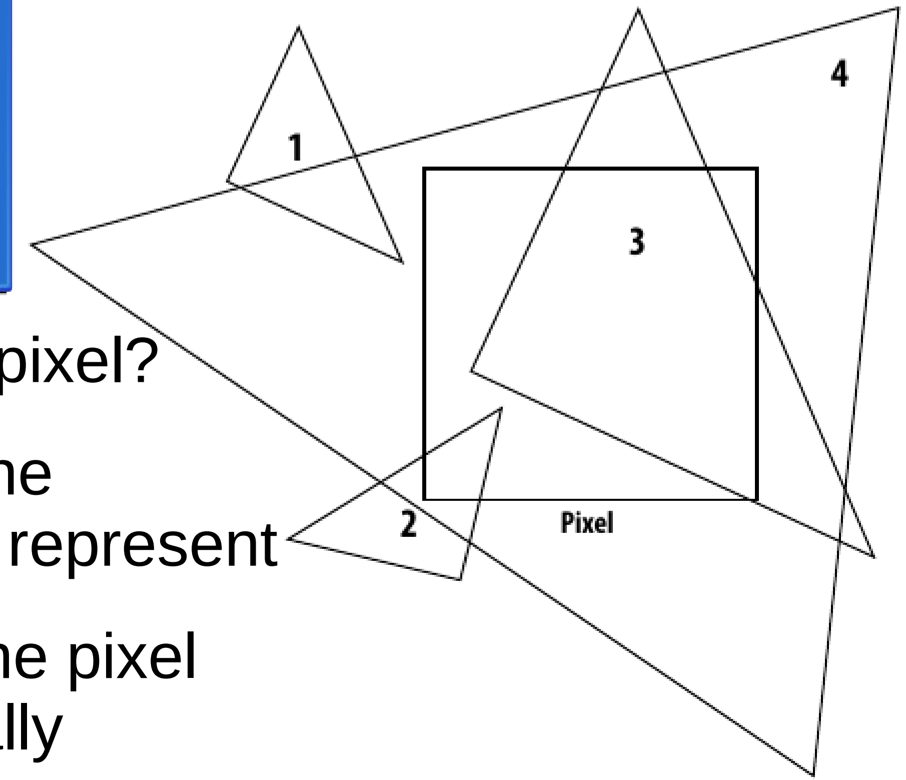
- We are currently looking at the stage called **rasterisation**, which is the task of taking an image described in a vector graphics format (shapes) – which we can obtain using projection - and converting it into a **raster image** (a series of pixels, dots or lines that when they come together on a display, they recreate the image).

Rasterisation or ray casting/tracing?

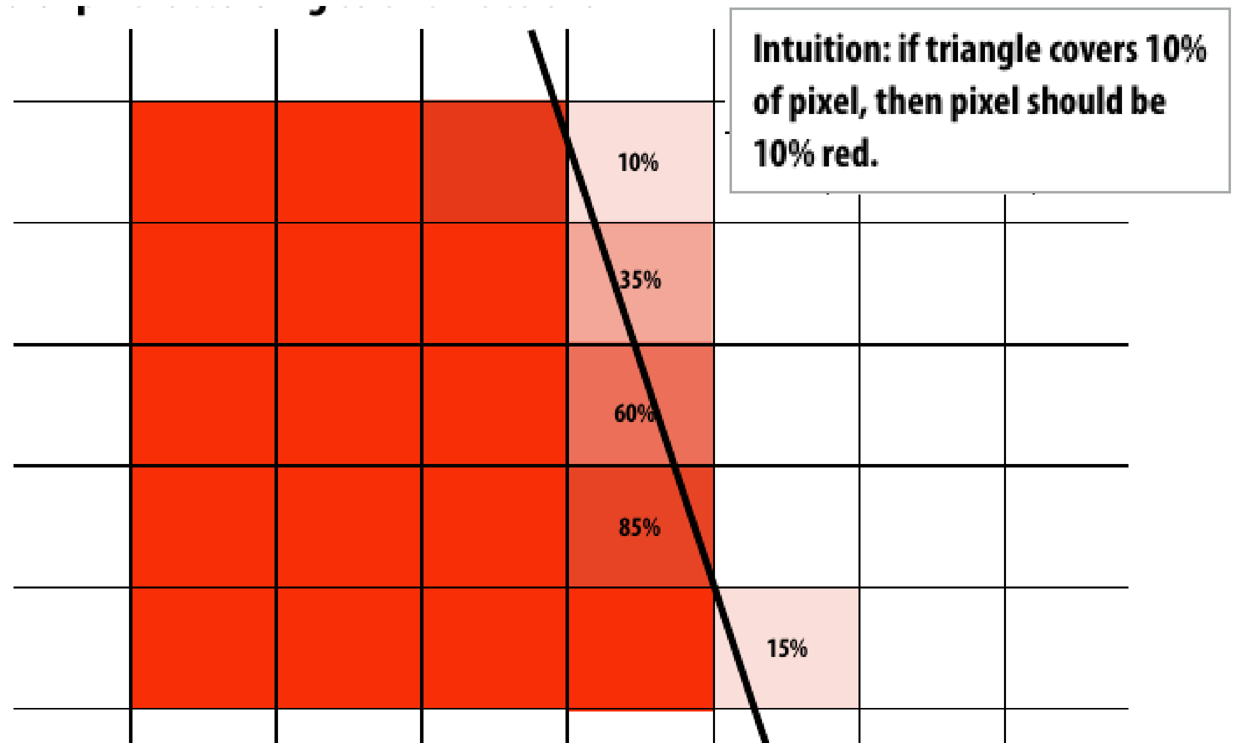


- Our transformation/projection chain allows us to methodically convert 3-D points to 2-D pixel locations
- But, in the class explanations, the metaphor of tracing light rays coming off/towards the objects is often in use; are we then using ray tracing?
- Rasterisation is considered an efficient and simple, albeit limited solution to **ray tracing**, which is supposed to mimic the behaviour of light and give high fidelity results

Coverage

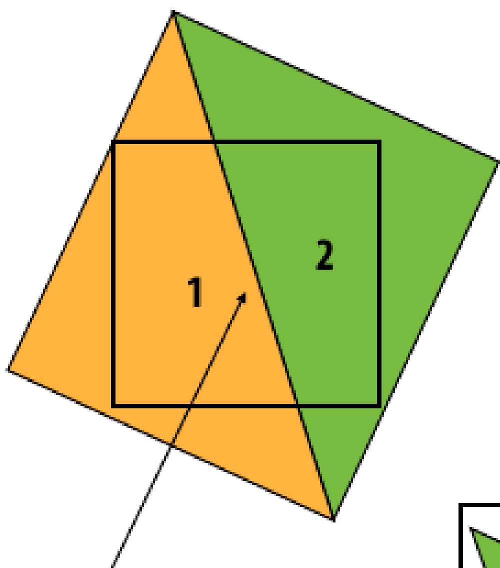


- Which of the triangles actually “cover” the square pixel?
- Our objective is to colour the pixels according to the (arbitrary) colour of the triangle which they should represent
- We could potentially measure the percentage of the pixel occupied by every shape and colour it proportionally

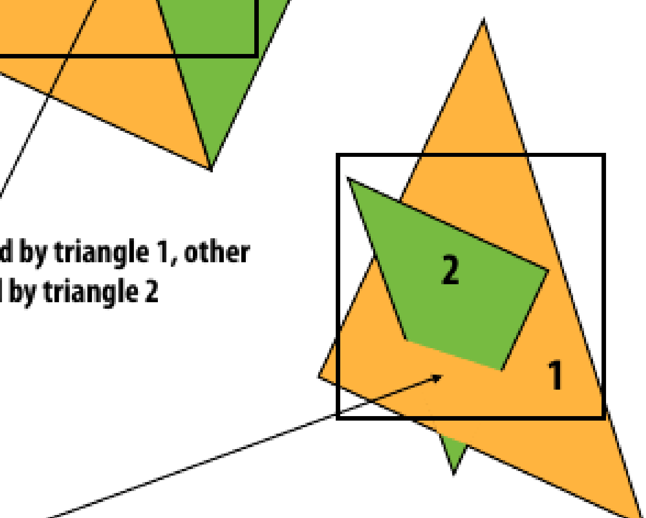


Coverage

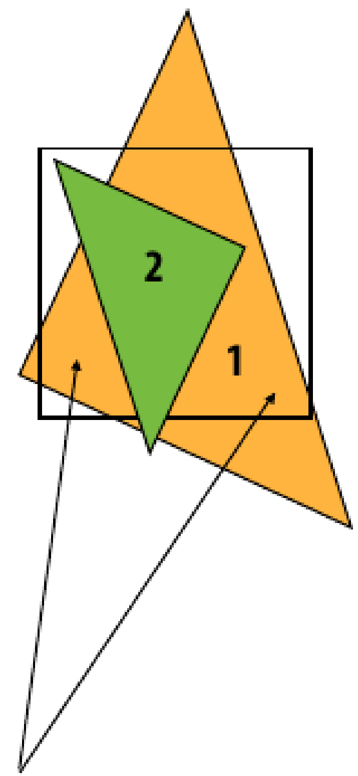
- Analytical schemas like this get much less convenient when we consider occlusions:



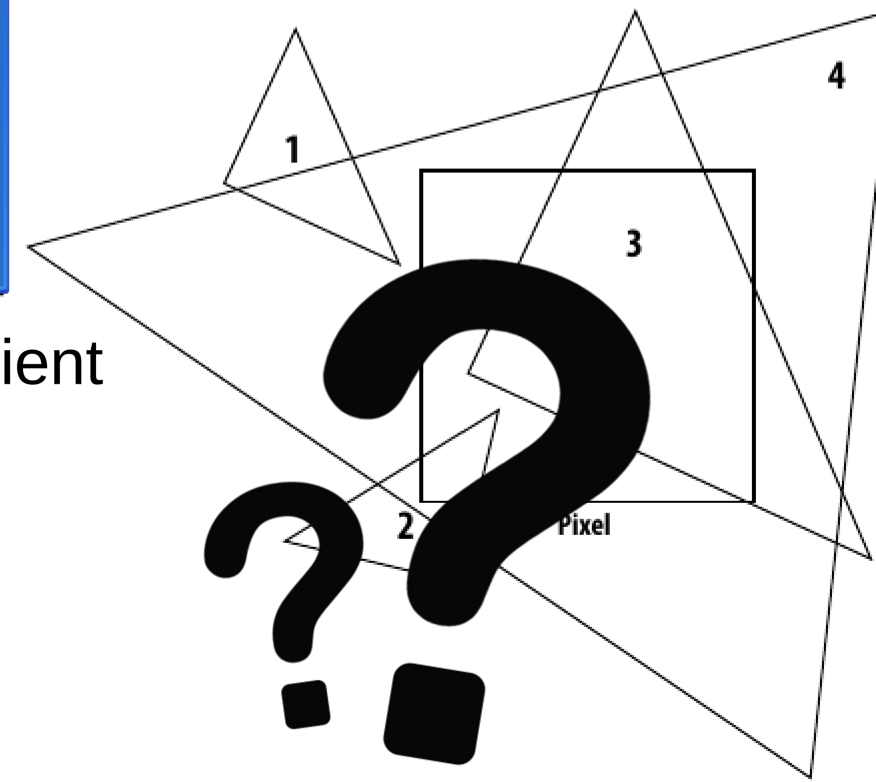
Pixel covered by triangle 1, other half covered by triangle 2



Interpenetration of triangles: even trickier



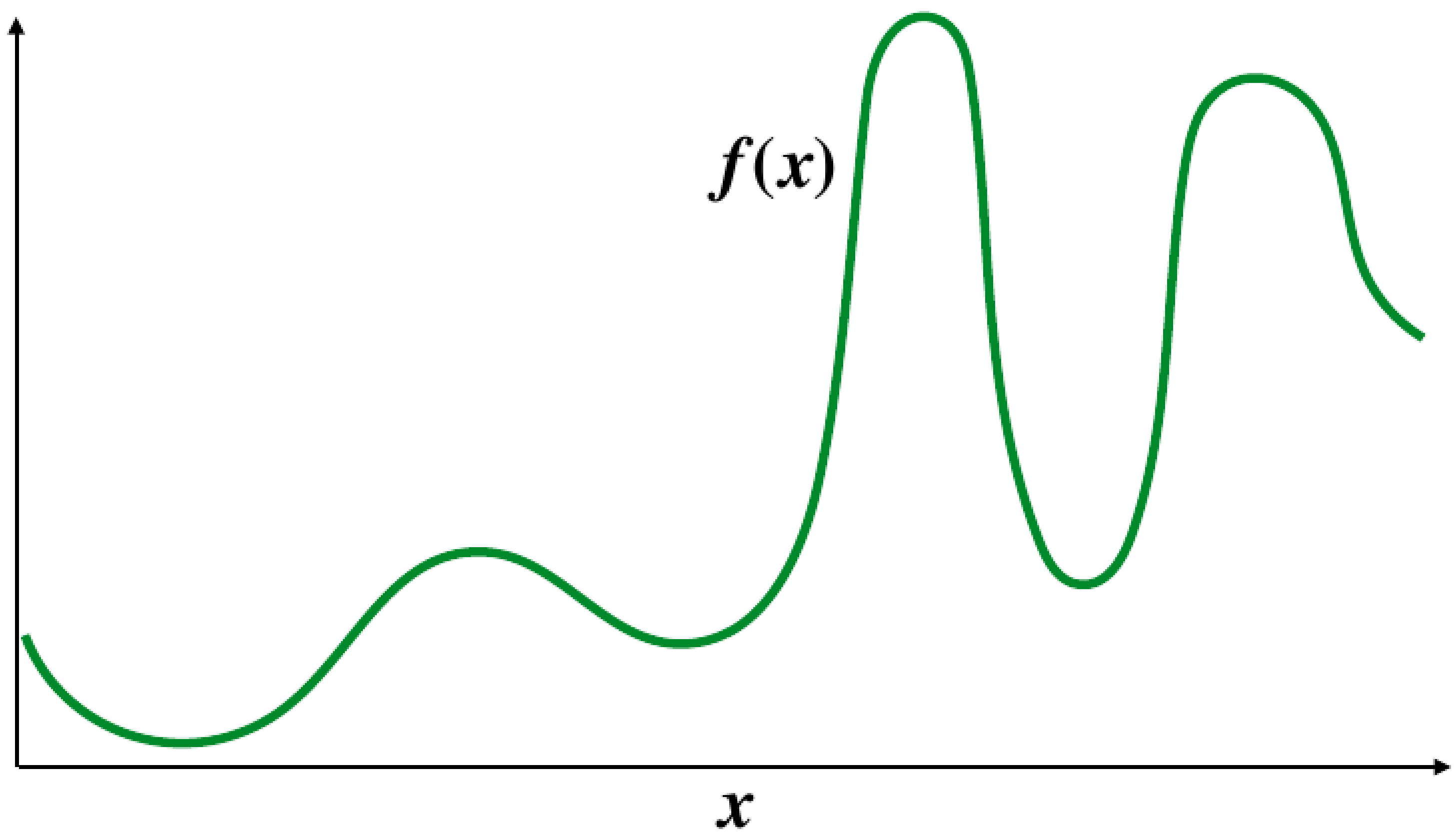
Two regions of triangle 1 contribute to pixel. One of these regions is not even convex.



At this stage, it is interesting to reflect on a good formulation of this problem in terms of sampling theory!

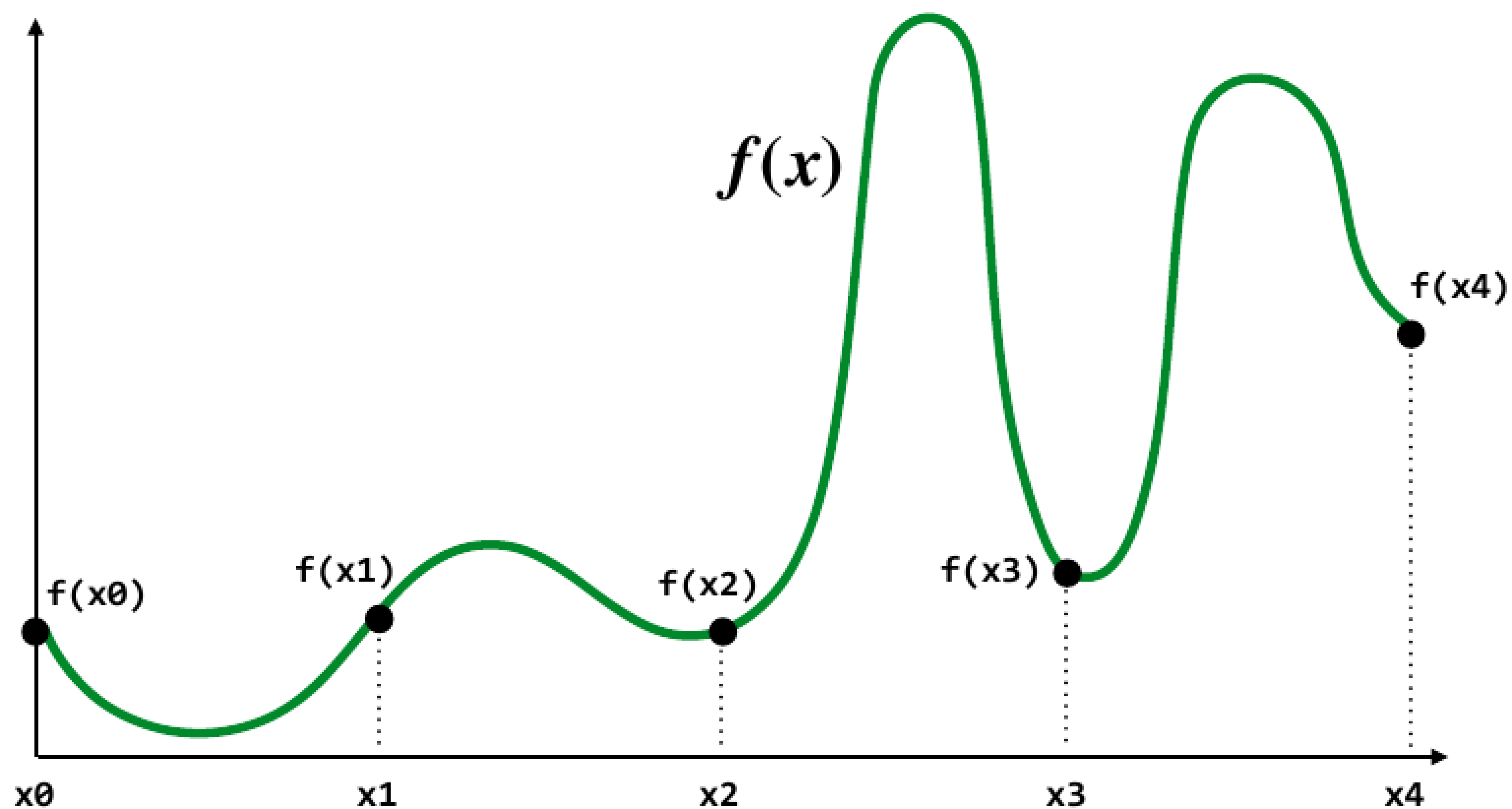
Sampling – backbone of rasterisation

- Let's start with a 1-D signal which we want to convert to a discrete format (=rasterise).



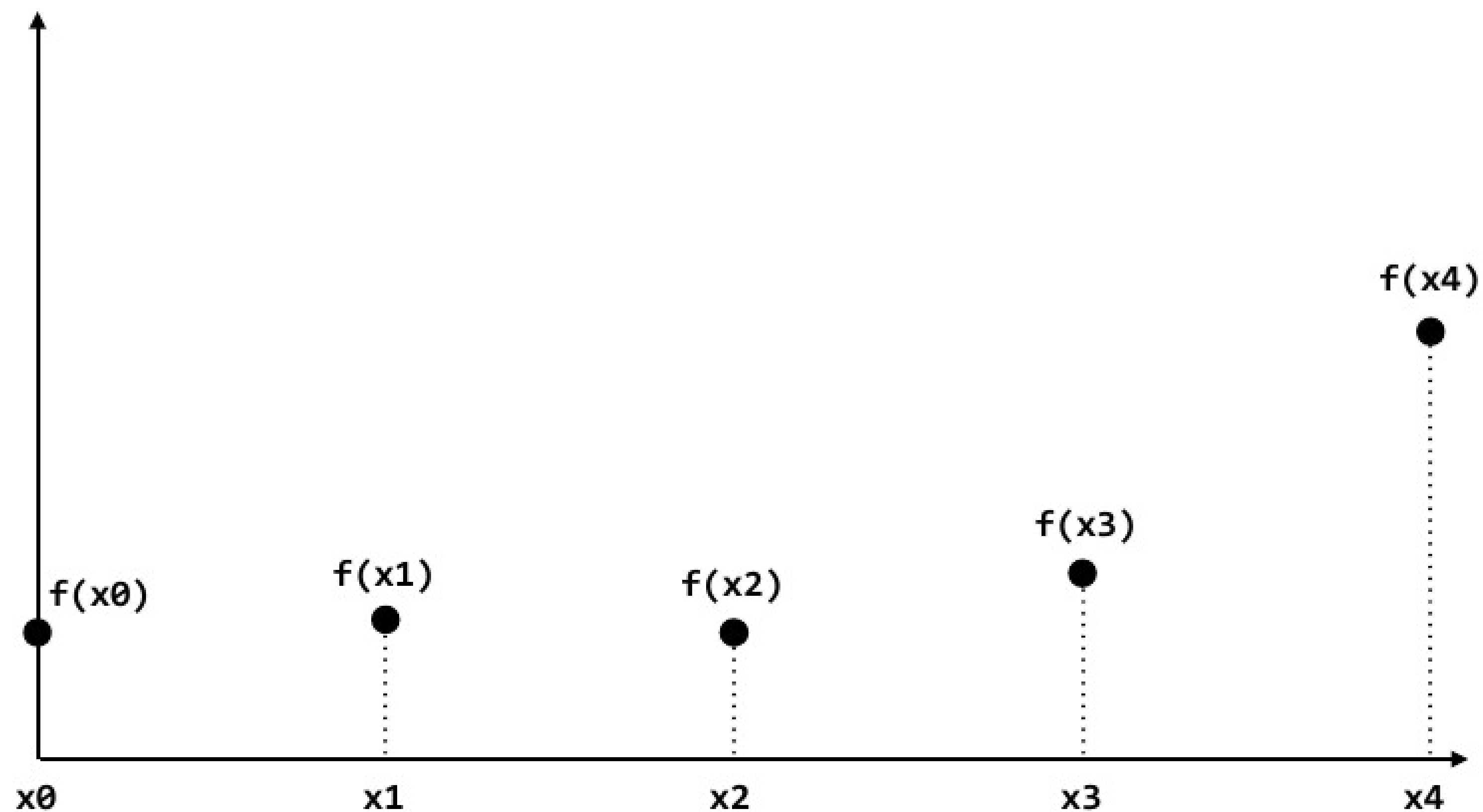
Sampling – backbone of rasterisation

- Sampling is expensive, so we are trying to limit the number of samples



Sampling – backbone of rasterisation

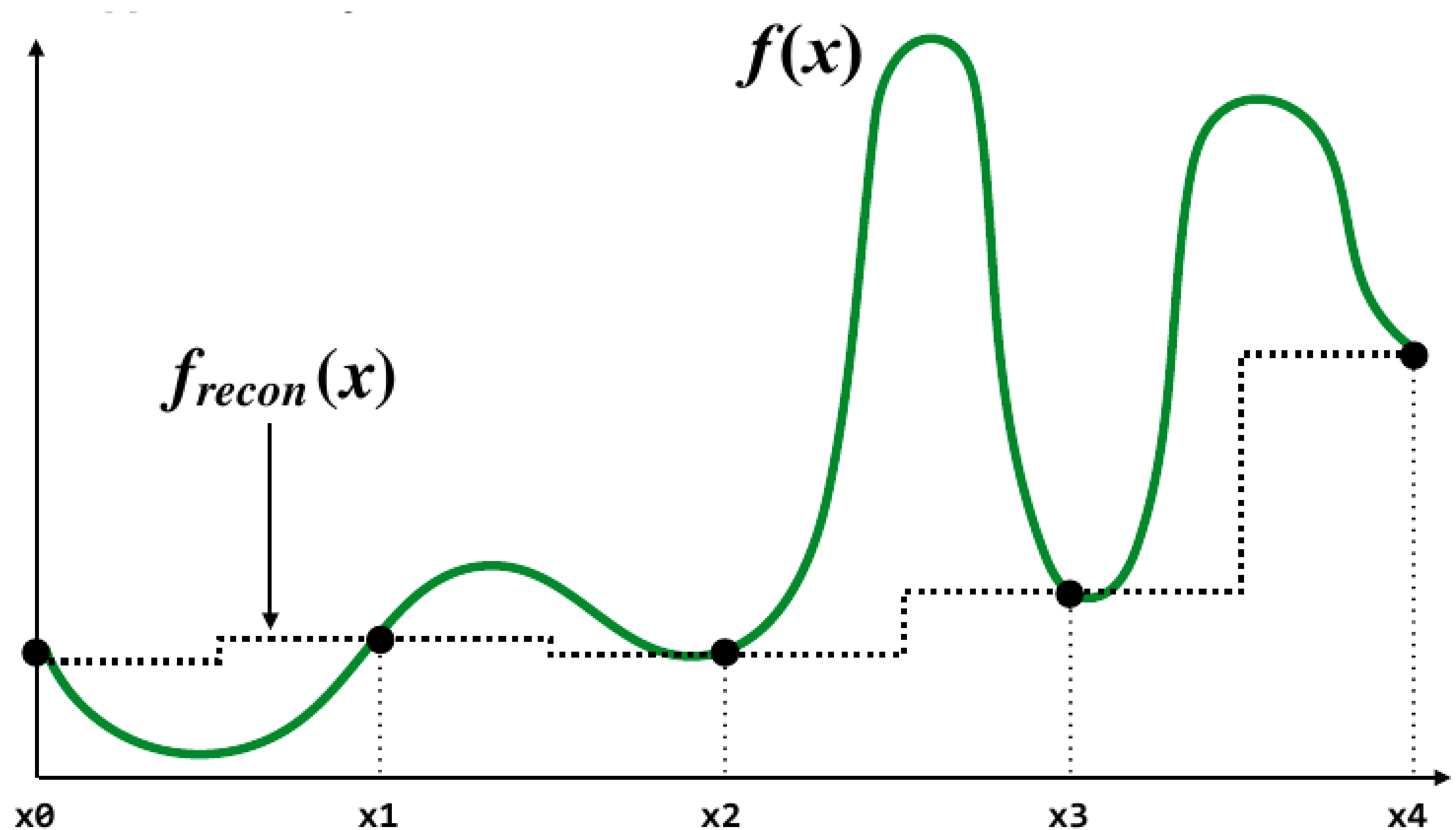
- Trouble starts when we want to reconstruct our signal...



Sampling theorem is a fundamental bridge between continuous-time signals and discrete-time signals. It establishes a sufficient condition for a sample rate that permits a discrete sequence of samples to capture all the information from a continuous-time signal of finite bandwidth. The sampling theorem introduces the concept of a sample rate that is sufficient for perfect fidelity for the class of functions that are bandlimited to a given bandwidth.

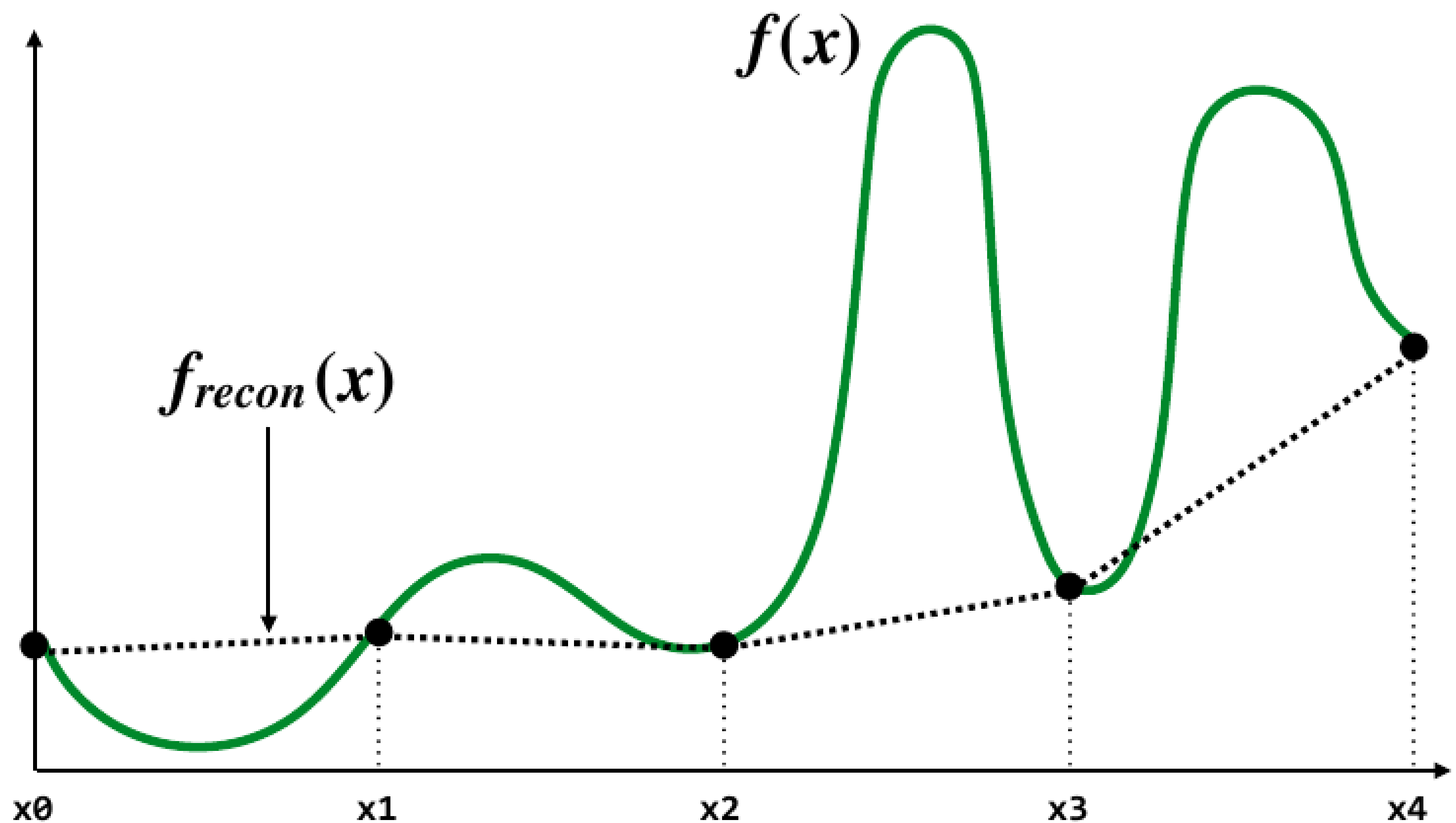
Sampling – backbone of rasterisation

- Reconstruction: piecewise **constant** approximation



Sampling – backbone of rasterisation

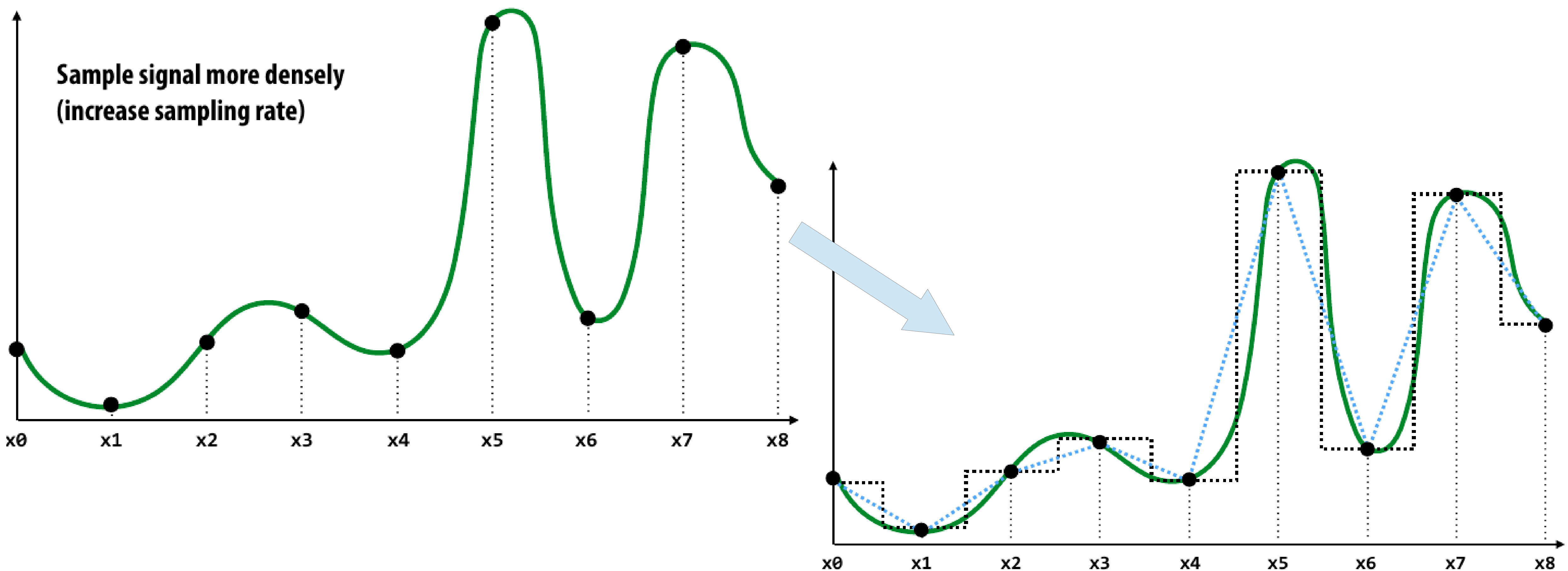
- Reconstruction: piecewise **linear** approximation



None of them can overcome the natural limits due to Nyquist frenquency!

Sampling – backbone of rasterisation

- Reconstruction: denser sampling can often improve the case



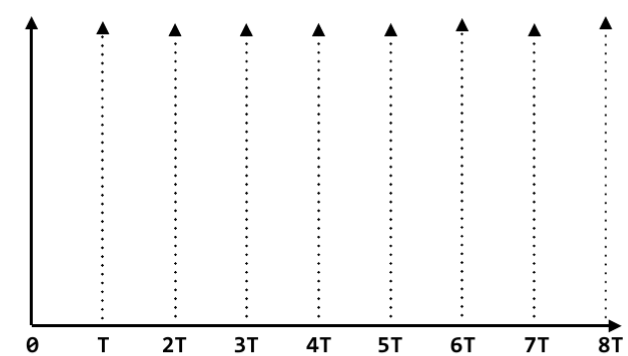
Sampling mathematics

- Dirac delta function $\delta(x)$ for $x \neq 0, \delta(x) = 0$ and $\int_{-\infty}^{\infty} \delta(x) dx = 1$
- We can use it to “pick” a value out of a continous function *f(x)*:

$$\int_{-\infty}^{\infty} f(x)\delta(x)dx = f(0)$$

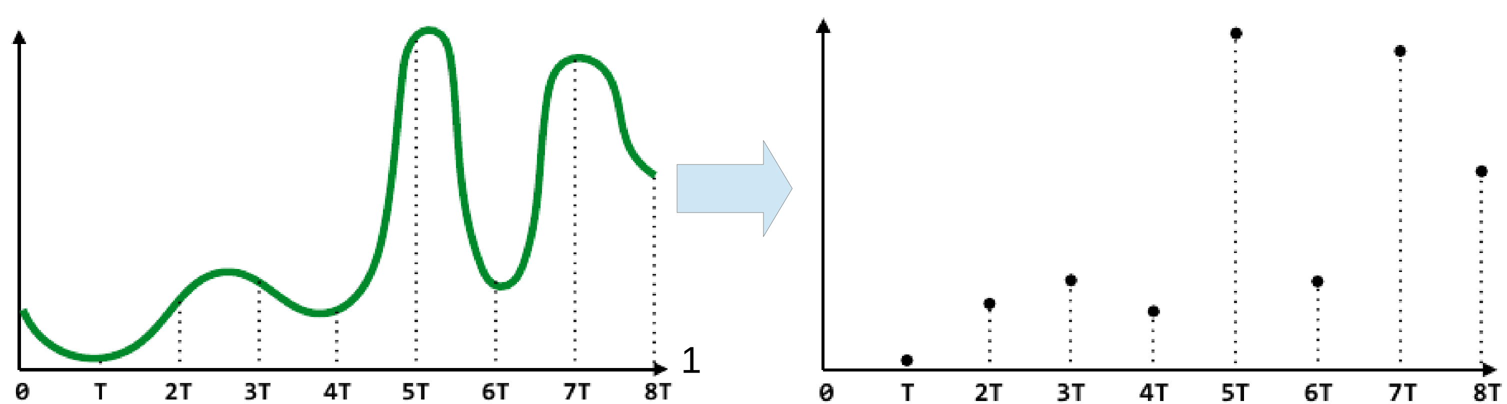
- We can construct a “comb” of shifted delta functions:

$$\text{Ш}_T(x) = T \sum_{i=-\infty}^{\infty} \delta(x - iT)$$



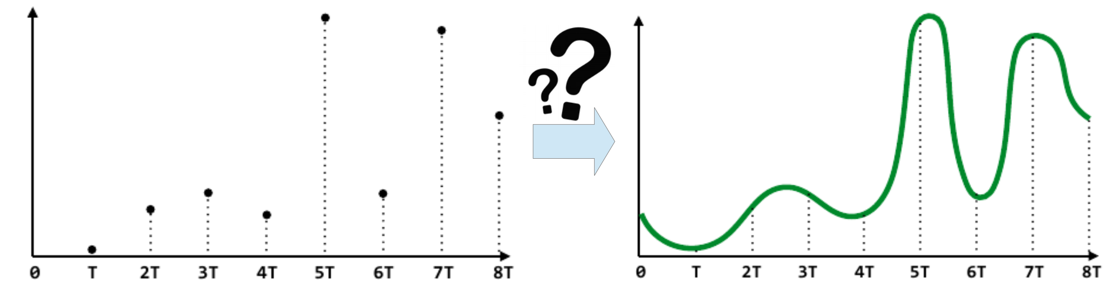
- ...and apply it to the function of interest:

$$\text{Ш}_T(x)f(x) = T \sum_{i=-\infty}^{\infty} f(iT)\delta(x - iT)$$



Sampling mathematics

- How do we handle reconstruction?

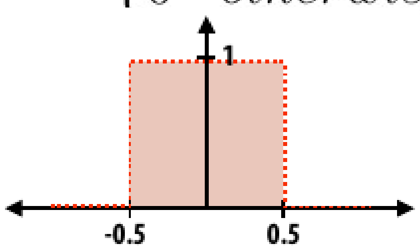


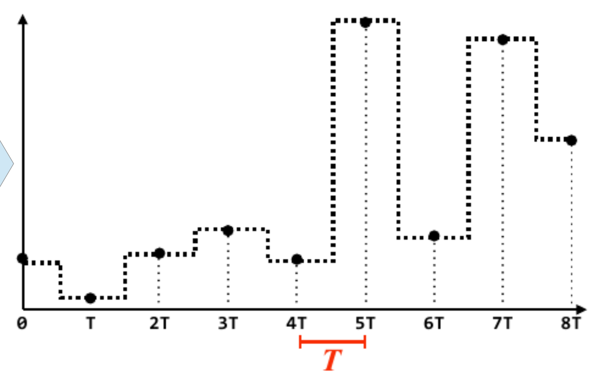
- Convolution

$$\underbrace{(f * g)(x)}_{\text{output signal}} = \int_{-\infty}^{\infty} \underbrace{f(y)}_{\text{filter}}\underbrace{g(x-y)}_{\text{input signal}}dy$$

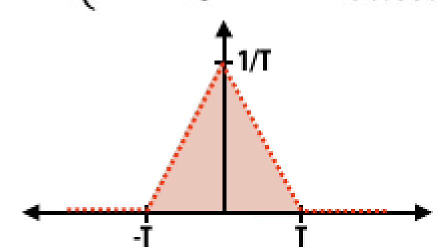
- The filter can be quite simple, e.g. a box or a triangle function

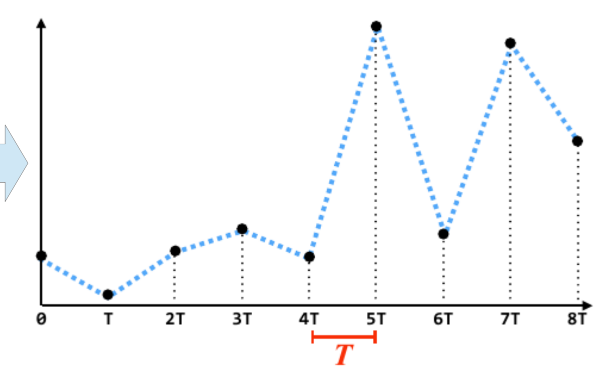
$$f(x) = \begin{cases} 1 & |x| \leq 0.5 \\ 0 & otherwise \end{cases}$$



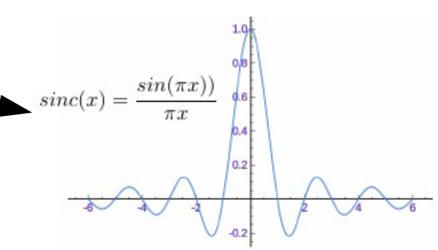


$$h(x) = \begin{cases} (1 - \frac{|x|}{T})/T & |x| \leq T \\ 0 & otherwise \end{cases}$$



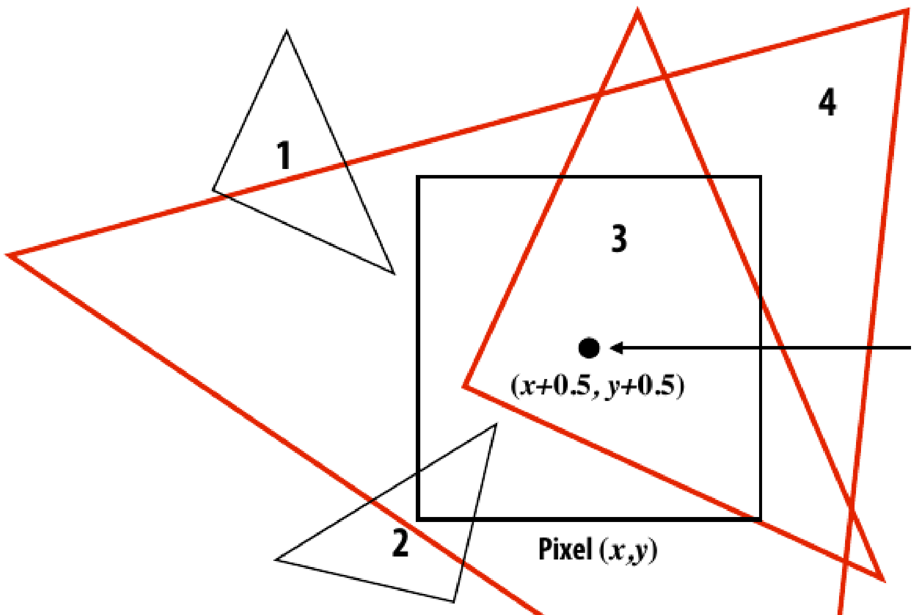


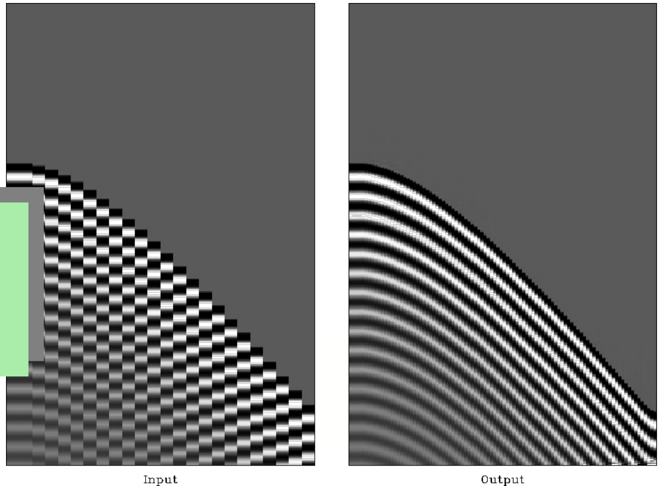
- In practice: normalised sinc filter, or similar



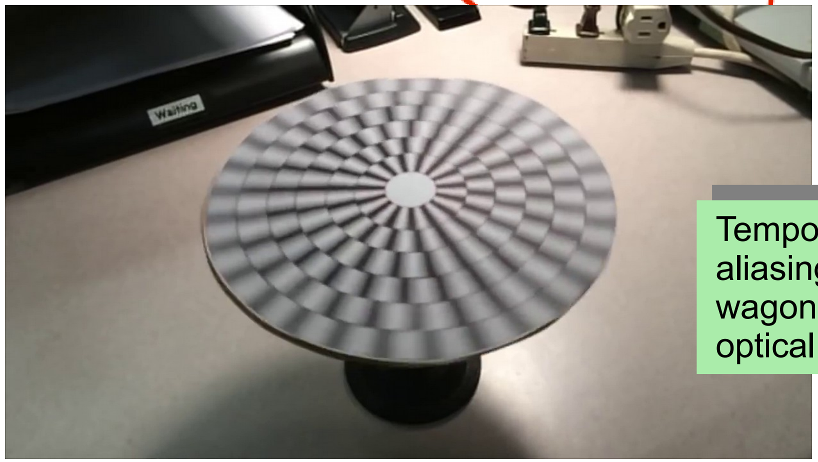
Sampling: motivation

- We can treat the question of coverage (= which triangle covers which pixel) as a 2-D function: coverage(x,y)
  - Pixels define our sampling period
  - Here: the pixel center coordinate are used to sample the triangles coverage
- Plus, sampling theory will teach us a couple of lessons:
  - Subsampling helps reconstruct the signal
  - Aliasing
    (and anti-aliasing!)



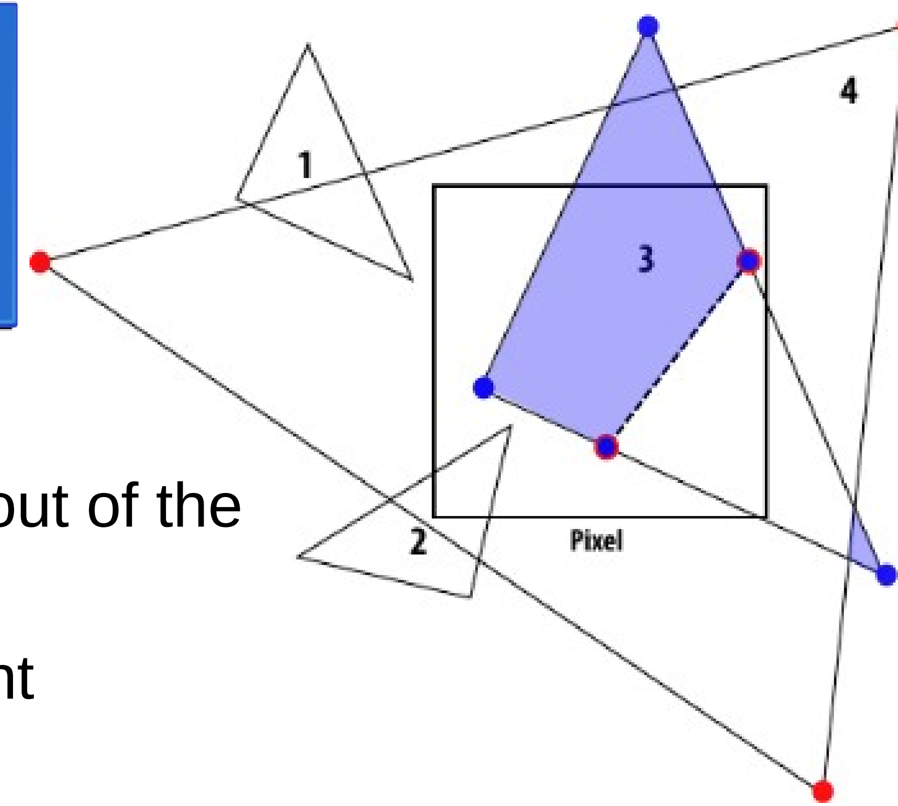


Aliasing in the left downsampled image – can you tell which way the lines are going?



Temporal aliasing – the wagon wheel optical illusion

Occlusion



- Which triangle is in front of the other one?
  - ray tracing reply: whichever is hit by a ray sent out of the camera first
  - rasterisation reply: we can calculate it, if we want
    - it’s an expensive operation, though
    - while doing screen projection for points in our shapes, we already have some z-depth information
    - It would be great to extend it to all points in the triangle – we would just need to interpolate between the vertices!
    - We can construct an image with this information for all pixels and keep track when new elements come → **z-buffer**

This is NOT an innocent remark! We will spend some time looking at this problem!

Thank you!

- Questions?