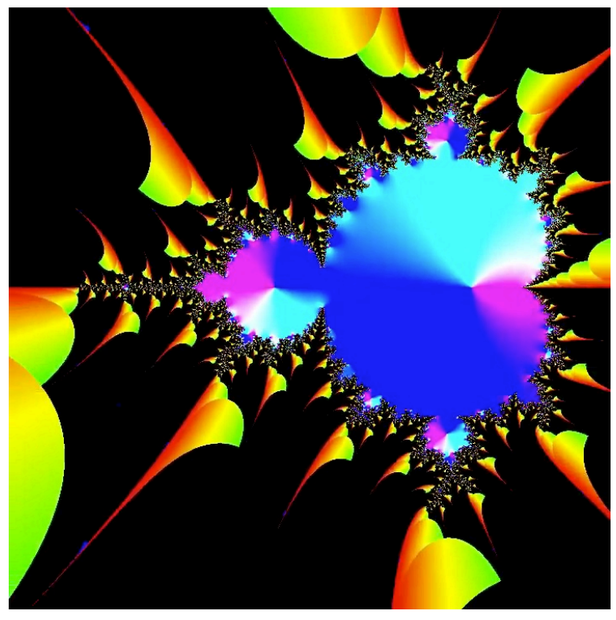
TSBK 07

Computer Graphics

Ingemar Ragnemalm, ISY



TSBK07 Computer Graphics

Spring 2017

Course leader/examiner/lecturer:
Ingemar Ragnemalm
ingis@isy.liu.se / ingemar@ragnemalm.se

Course home page:
http://www.computer-graphics.se/TSBK07.html

This lecture:

Course plan and overview

The course subject

Projects, some demos

Graphics systems

Who am I?

- **Lecturer/associate professor**
- **Researcher**
- **Game & graphics programmer**
- **Game designer**
- **Hacker**

(plus runs in the forest looking for tupperware)

Who are you?

- **4-5th year on 5 year program**
- **International students**

MOST of you have no prior experience of CG
SOME of you have some experience of CG
A FEW of you have extensive experience of CG

All of you know some programming. (Some are “wizards”)

People:

Examiner/lecturer:
Ingemar Ragnemalm

Lab assistants:
Jens Ogniewski
Marcus Wallenberg

Image-related technologies:

Image Processing

Image Editing

Computer Graphics

Image Analysis

Computer Vision

Image Coding

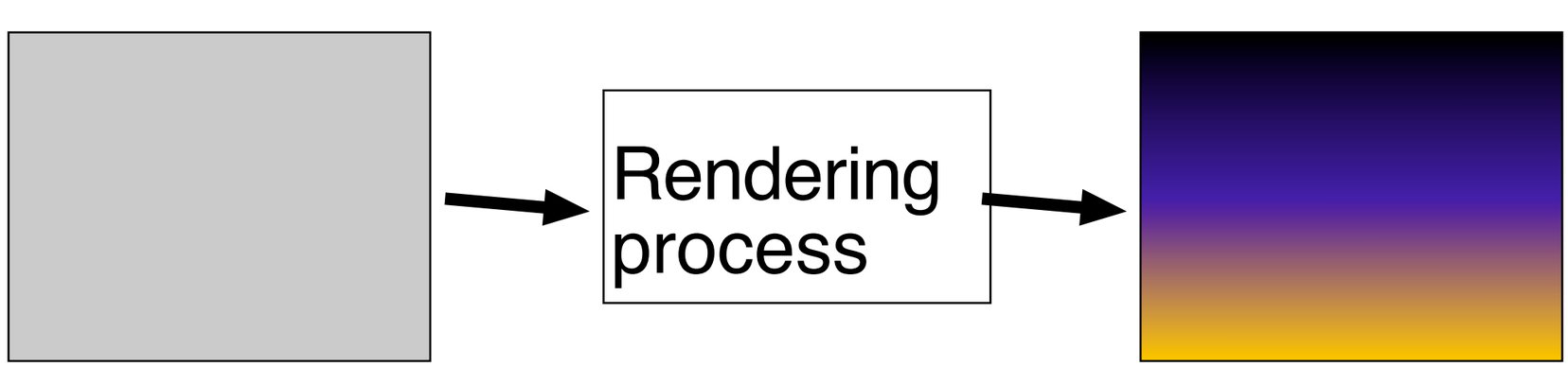
Image Compression

Graphic Arts

Tomography

etc…

Computer Graphics



Description

Image

Computer Graphics:

Creating images from non-image data

Cube =
(10, 10, 10)
(10, 10, 20)
(10, 20, 10)
(10, 20, 20)
(20, 10, 10)
(20, 10, 20)
(20, 20, 10)
(20, 20, 20)
Camera =
(80, 60, 60) (15,15,15)

Learning goals:

You should get:

- **Experience of computer graphics programming (OpenGL and shader programming)**
- **Learn important concepts and methods for implementing computer graphics applicatons**
- **Understanding of some important low-level graphics algorithms**

Everybody should learn something new!

Course book:

Note: *Blue frame* version only!

(Modern OpenGL version.)

Available on-line as PDF!

Time schedule

VT1:

Lectures (15)
Labs (4)
Project specifications

VT2:

Project work
Project demonstrations
Reports
Lessons (2)
Written exam

Lecture plan

1. Introduction, graphics systems, API's

2. 2D graphics, OpenGL/GLSL introduction
3. 3D transforms, viewing, projection
4. Introductiom to visible surface detection, light models, shading
5. Surface detail, texture mapping

6. More surface detail, more VSD
7. Rotation around arbitrary axis, normal matrix, multitexturing

8. Large worlds, high-level VSD, level of detail
9. Billboards, curves and surfaces
10. More curves and surfaces, geometry shaders, animation
11. Collision detection, fractals and noise
12. Anti-aliasing

13. Ray-tracing and radiosity
14. Line drawing algorithms, polygon rendering
15. Loose ends, outlooks

Some minor changes are likely!

Laborations

Lab 1: Intro to modern OpenGL with GLSL
Lab 2: Models, camera, virtual world
Lab 3: Virtual world, advanced lighting with GLSL
Lab 4: Terrain rendering

Projects

Small projects! ≈ 1.5 week of full time work.

Lessons

2 lessons at the end, preparing for the exam

Projects:

Define a course-related problem to go deeper into.

2-4 people in each group.

Consider the project now and then during lectures and labs

Project suggestions should be handed in at the end of VT1!

Implement in VT2

Allocate time!

How to form a group

Balance: Avoid groups with dramatic difference in experience!

Size: Larger groups need more project management

If a group doesn’t work, let me know. Better split/ reform early than late.

One-person projects allowed - but not recommended

Resources:

It is allowed to use any material that you come across, i.e. on the web.

It is not allowed to do that without documenting it!

You must tell what you use and what you have added yourself!

Rule of thumb:

”Cheating is allowed as long as it is *properly documented*”

- Proper references to anything you didn’t do yourself.
This makes your results *stronger*, not weaker!

- Project report in your own words - no copy/paste!

- Reuse is not for free. A small project without external modules is easier.

Project examination:

- **Short presentation (3-4 groups at a time)**
- **Demonstration in the lab or on some other suitable computer**
- **Written report**

This should take place in may, with time left after it for the final work for the written exam.

Project ideas:

- **3D labyrinth**
- **Interactive solar system**
- **Robot with moveable limbs**
- **Driving simulator**
- **Flight simulator**
- **Enhanced terrain rendering**
- **Large virtual reality with frustum culling/level of detail/ portals/etc**

Advanced project ideas:

- **Demo with several scenes with fun effects.**
- **Working game with some nice visuals.**
- **Game/VR with advanced collision detection/handling**
- **Terrains with geomipmapping**
- **Particle systems**
- **Fractal vegetation**
- **Advanced light effects**
- **That neat, advanced effect that I always wanted to do.**

But don’t aim too high!

Different people have different style...

All project should not be similar!

- **Some focus on specific algorithms**
- **Some are “applications”, like games**
- **Some are artistic, neat “demos”**

Do it *your* way!

Grades

Your grade is determined by the written exam!

...but...

Particularly good projects may qualify for bonus points on the exam!

- More complex

- Polished, “finished products”

- Based on current research

The bonus can not help you pass the exam, but may give you a higher grade! Must be applied for, not automatic.

Level of bonus: 5 points (1/2 grade step)
10 points (a full step) in extreme cases

The lab material is available on the course home page:

http://www.computer-graphics/TSBK07.html

Lab location:

Egypten, Asgård, Olympen:
Entrance 25, upper floor

Southfork:
Entrance 27, upper floor, corridor C

Recommended lab: Southfork!
Olympen is OK (but old).
The others are rubbish!

Late changes in the lab material are possible.

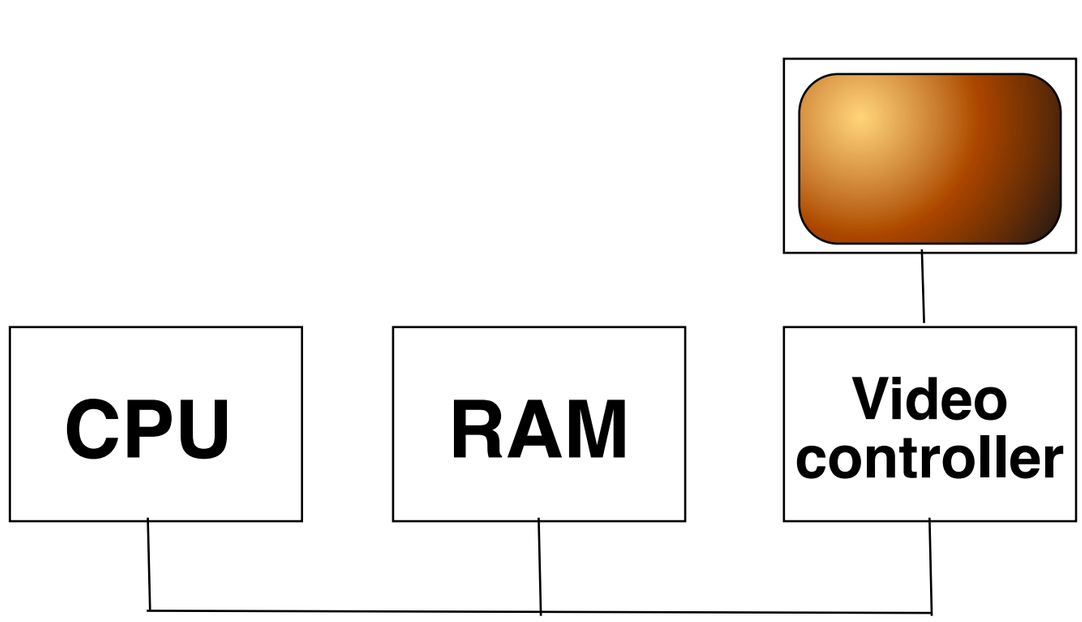
The current lab material should be perfectly usable

but

the deadline for the lab material is at the start of the lab, so I give myself the right to make changes.

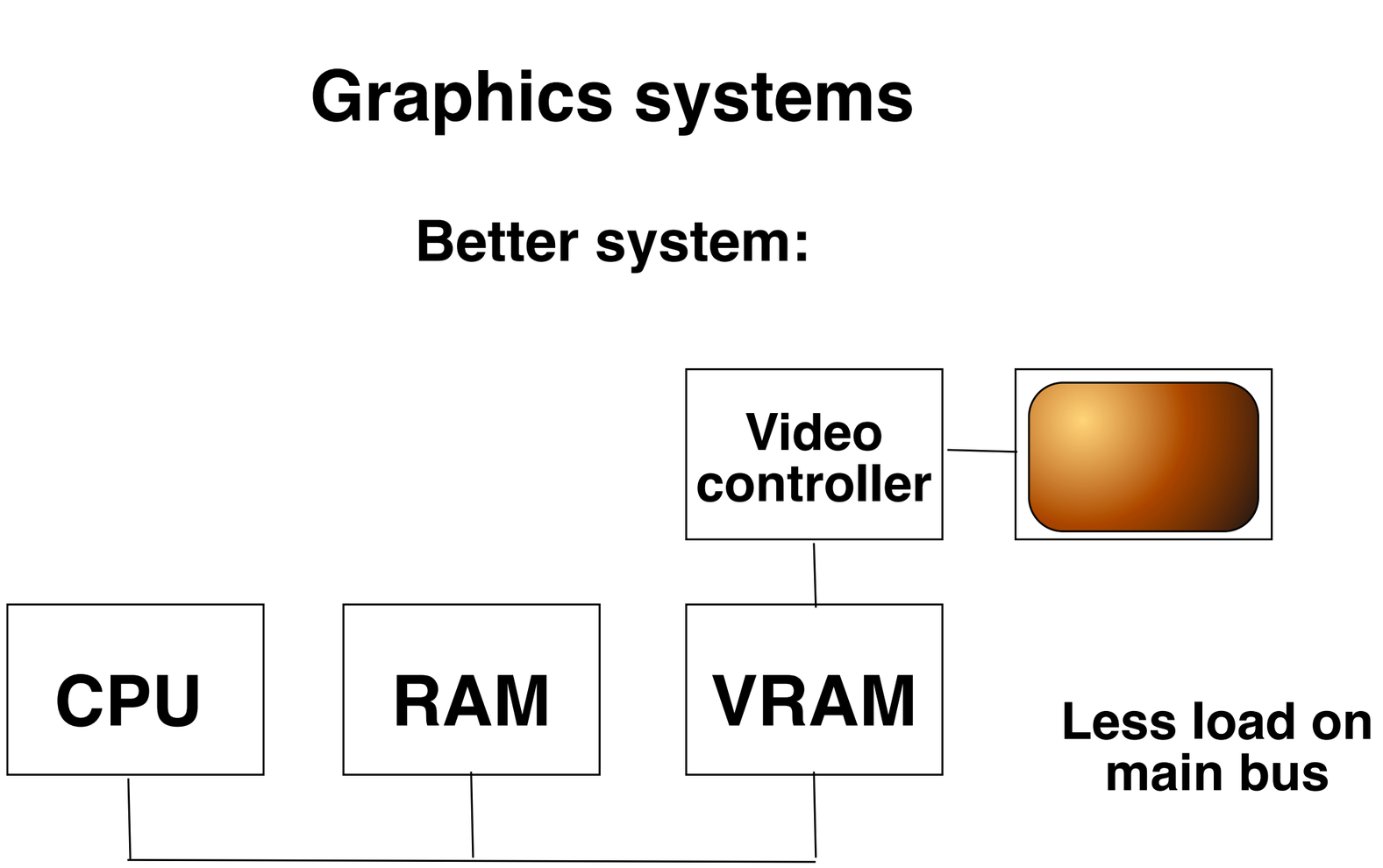
Graphics systems

Simple system:



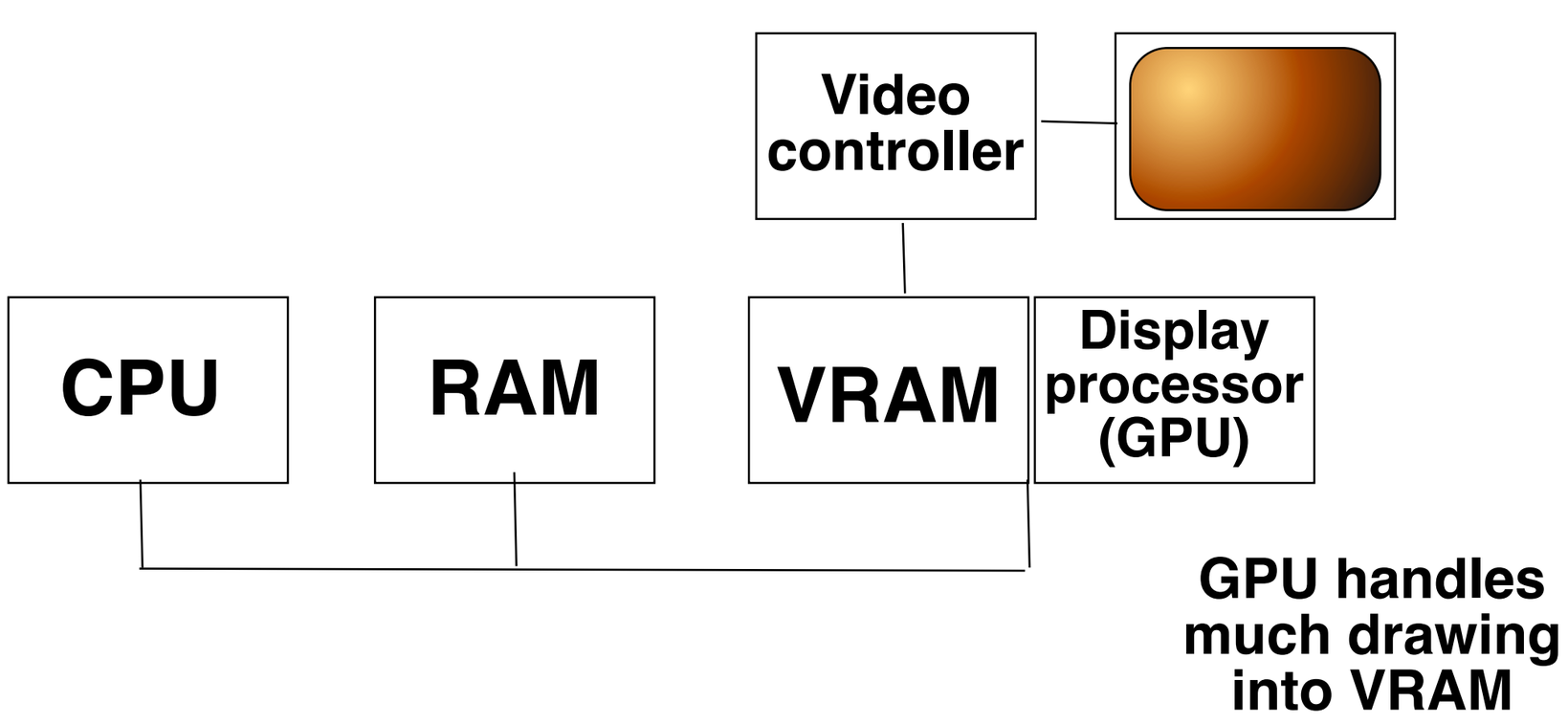
Graphics systems

Better system:



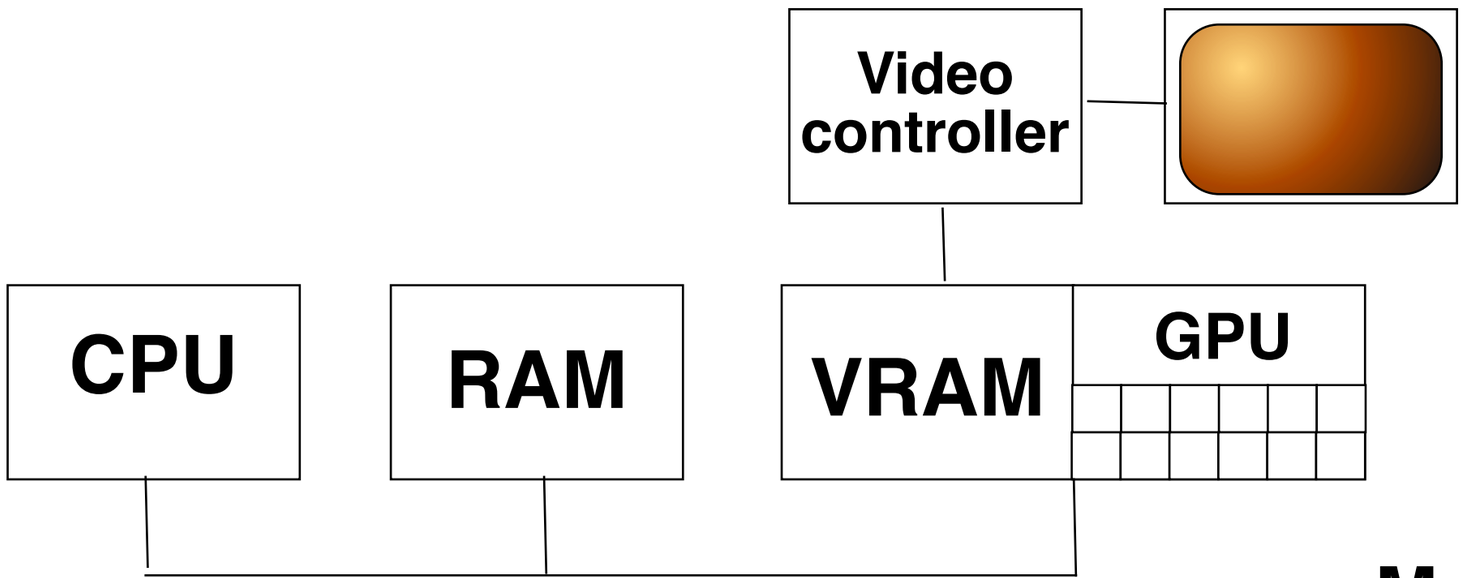
Graphics systems

Even better system:



Graphics systems

Current system:



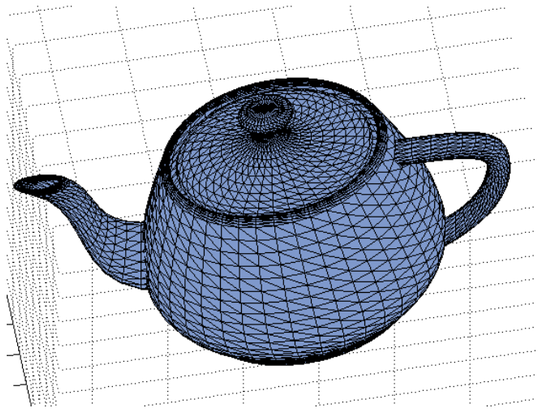
Multi-core, programmable GPU!

What is an image?

What is an image?

An array of pixels!

What is a pixel?



What is an image?

Array of pixels

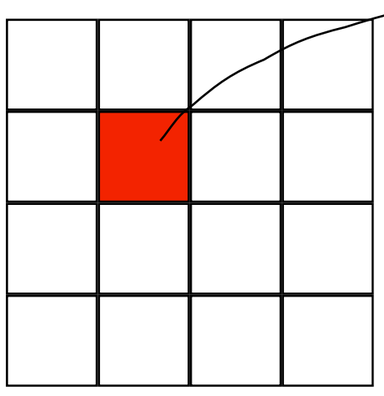
What is a pixel?

***Picture element*. Sample of a 2D signal.**

Image = array of pixels?

Pixel = one chunk of memory (usually)

”Chunky pixels”.



```
0ac807ffa0c860ff890102ff0c801d7fd0c90eff870b01ff5790s7ffe08730ff
4c7b94ff0s127eff741206ff6d562eff475665ff3e6141ff2121a2ff244d57ff
8c6c79ff8b7897ffd7e54aff4232e5ff4e6576ff789d78ff97d98cff798b78ff
97a987ffe56e67ff4563d4ffa675e3ff123112ffed1eceff1490a7ff5ad8adff
987bc8ff9b98c6ff656745ff646769ff812912ff879632ff189101ff5010c0ff
```

How big chunk?

How many bits?

Pixel size	# of colors	Usage
1	2	Masks
2	4	?
4	16	?
8	256	Safe mode
16	Thousands	”High color”
24/32	Millions	”True color”
Up to 256	Billions	FP color

What is color?

Colors

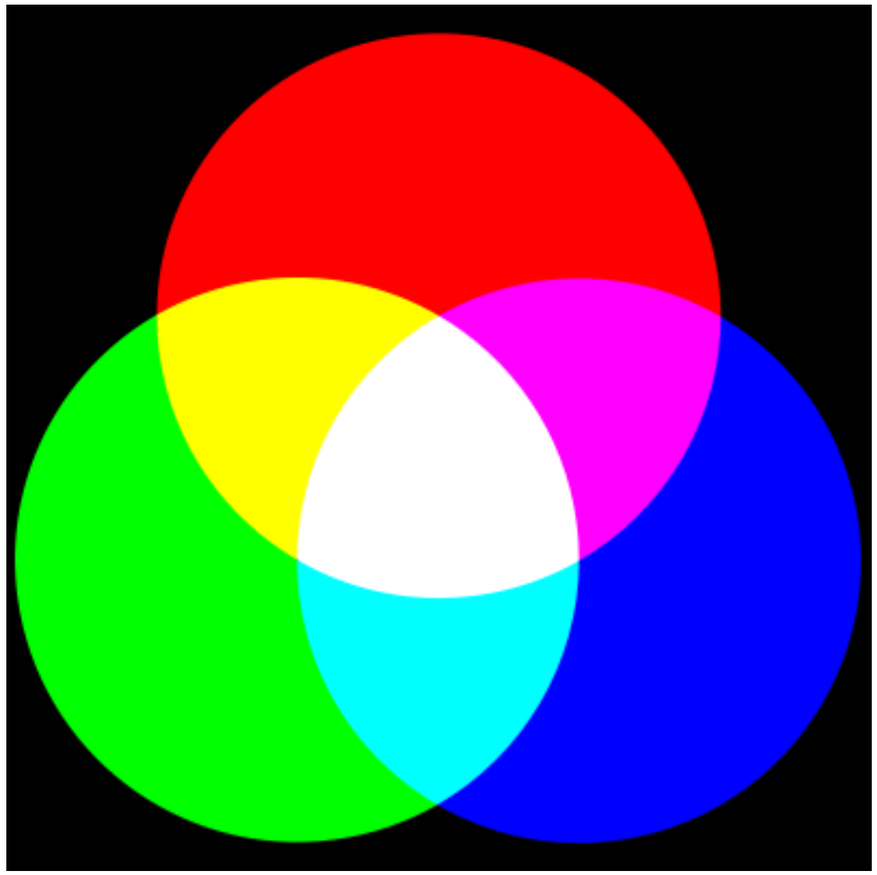
- beyond Bamse

Additive colors

Red
Green
Blue

Corresponds to receptors in the eye!

Used on color displays

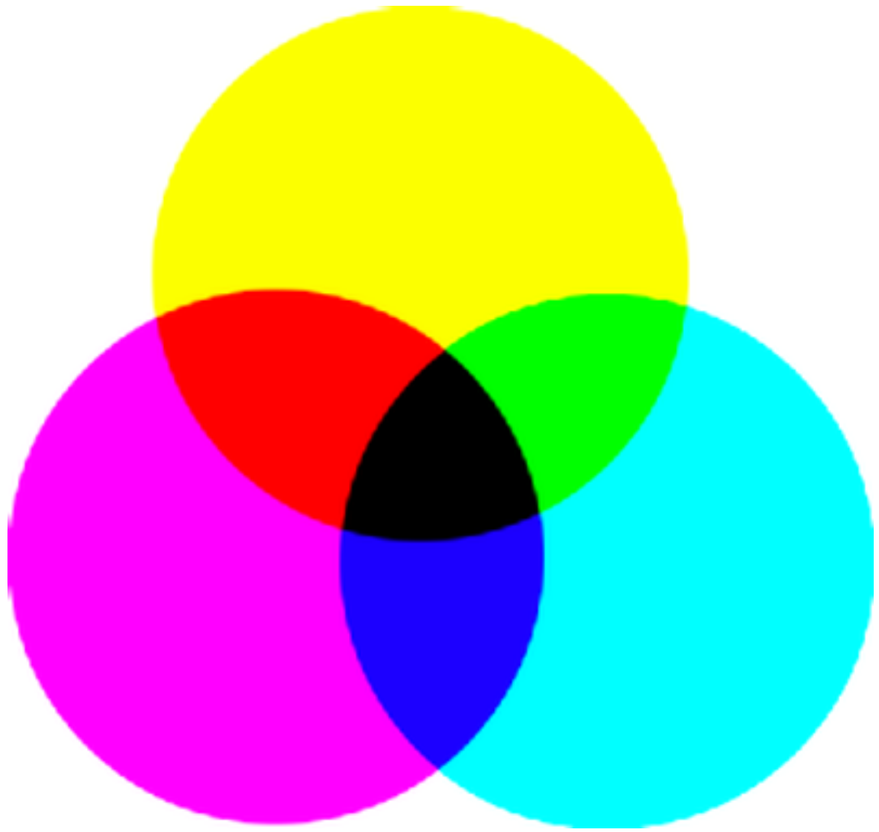


Subtractive colors

Cyan
Magenta
Yellow

Filters away primary colors

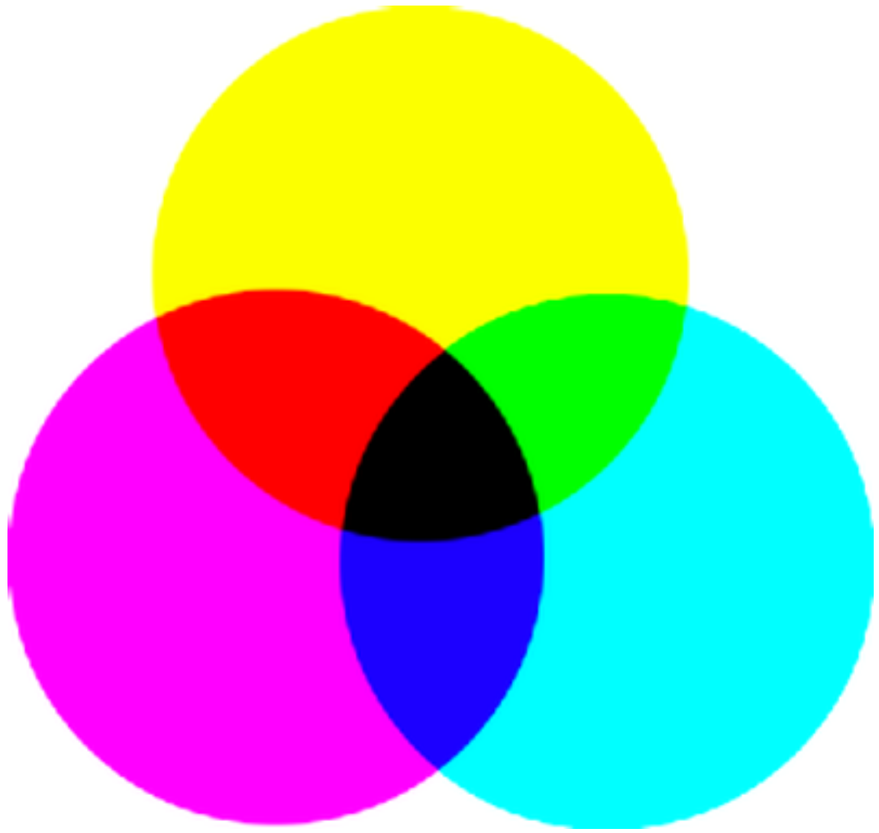
Used in printers



Subtractive colors

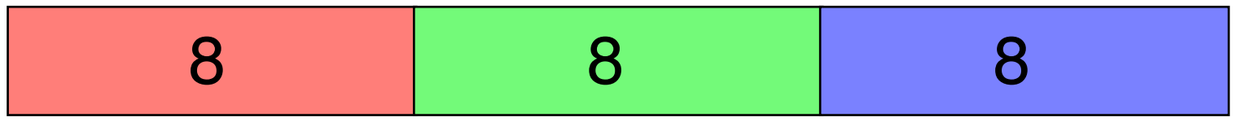
C+M+Y colors usually not perfect enough for a good black!

Printers often use CMYK = CMY plus black!



Colors in pixel values

Example: 24-bit color



Typical: 8 bits per channel (256 shades)

24-bit color

3 bytes per pixel bad for addressing efficiency!

Add fourth byte for getting a power of 2 size.

The fourth byte is often used for transparency - the *alpha channel*

RGBA color!

Colors in pixel values

Example: 16-bit color

5	5	5	1
6	6	4	-

Note that a lower number of bits can be used for blue. Why?

More to say about colors

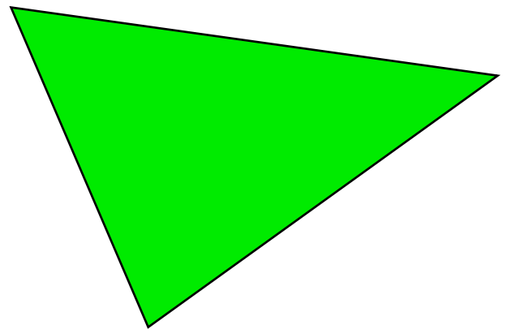
- XYZ color system
- Eye physiology
- Receptor ranges and sensitivity
- Perceptual issues

but this is enough for our purposes

The RGBA system is the important thing!

Computer graphics program:

- **Main program on CPU**
- **Send primitives (e.g. triangles) to GPU**
- **Shaders, kernels on GPU**



How do I code this?

Use the source, Luke!

Next time:

2D graphics

2D transformations

Algebra for computer graphics

Introduction to OpenGL