

Announcements

- **Please put your Term Paper on the table.**
(Late papers will be accepted Friday with 20 point penalty)
- **Midterm on Monday**

Color Theory

There are 16,777,216 colors to choose from!

How many should you use?

- **Too many is disruptive/confusing**
- **Too few could be boring**

Color Theory

Emotional Responses:

Red – strength, passion, energy, excitement

Orange – similar to red, but less aggressive (more “cheerful”)

Yellow – refreshment, energy

Green – nature, health, well-being

Blue – calm, peace, stability, trust

Purple – sophistication, spirituality

White – purity, trust

Black – depth, power, steadiness

Cool/Warm

Cool Colors: Green, Blue, Violet

- **Appear distant**
- **Great for backgrounds**

Warm Colors: Red, orange, yellow

- **Appear closer-up**
- **Great for menus**

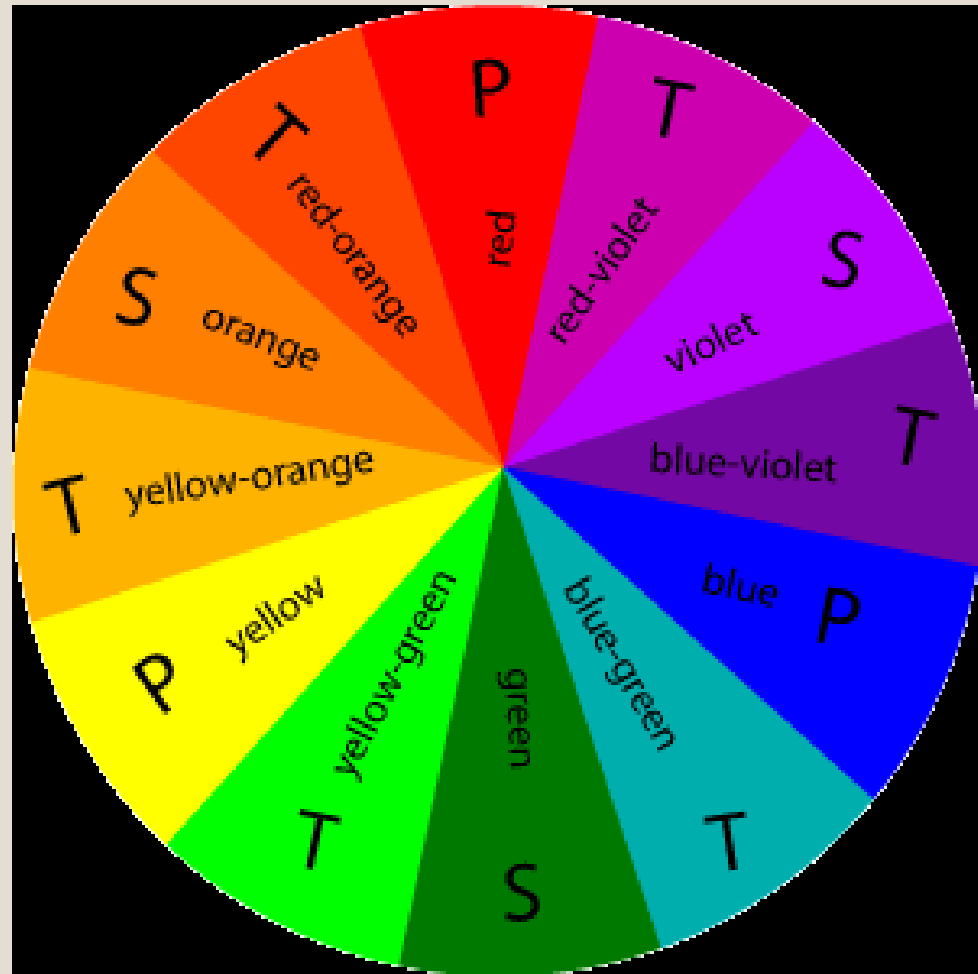
Standard Color Wheel

(P) Primary

(S) Secondary

(T) Tertiary

...



Color Wheel Definitions

Complimentary Colors – opposite on wheel

- **Highest contrasts**

Analogous Colors – close to each other

- **Lowest contrast**

Harmonic colors – equally spaced on wheel

- **“Harmonic dyad” (complimentary colors)**
- **“Harmonic triad”**
- **Appealing to the eye**

Color Strategies

1. Main/Support/Emphasis colors

- **Main color**
 - Occupies most of the page
 - Sets the tone
- **Support color**
 - Analogous to main color
 - Used to surround main color
- **Emphasis color**
 - Complementary of main
 - Used to highlight specific elements

Color Strategies

2. Monochromatic

- One main color
- Entire page is variations of darker/lighter versions of this color and analogous colors

Color Strategies

3. Harmonic Triad

- **Three equidistant colors from wheel**
- **Looks balanced**

Examples

Some select pages with great color schemes:

<http://inspiredm.com/colour-schemes/>

Some tools for creating color schemes:

<http://paletton.com/>

Top 10 Ways to Become a Better Programmer

Programming Tips

10. Be confident!

- Everyone struggles at first
- Anxiety is *normal* for beginners
- A large project can seem overwhelming... Break it into steps
- It takes practice before it becomes second nature
- Be patient!

Programming Tips

9. Don't Procrastinate

- It is **impossible** to know how long something will take!
- It doesn't matter how good you are

Programming Tips

8. KEEP BACKUPS!!!!

Programming Tips

7. Plan before you begin!

- **Time spent planning more than pays for itself**
- **Without a plan you are likely to reach many “dead-ends”**
- **How can you break the problem into manageable pieces?**
- **Write down some pseudocode and/or draw some diagrams (See next two slides about pseudocode...)**

“PseudoCode”

- **Pseudocode – halfway between English and Code**
 - Mostly English words
 - Variables frequently used
 - Structured like a program
 - Ignores formal language rules
 - Does not depend on a particular programming language
- **Useful for jotting down the flow of a program without having to worry about all the technical details of formal programming**

Example: I would like to write a program that sends an email message. The message can be sent to just one recipient or everyone in the user’s address book. (Next slide, please...)

PseudoCode Example

prompt: “Enter message”

input message

prompt: “Send to entire address book?”

input response

if response is no

prompt: “Enter recipient”

input recipient

send message to recipient

otherwise

for each address, x, in the address book

send message to x

Programming Tips

6. Don't make assumptions

- If you don't know how something works, look it up!
- Never assume the user will do what is “expected”
- Never just assume that you did the *easy* part correctly – we all make dumb mistakes sometimes!

Programming Tips

5. Learn to debug your code

- Be systematic
- Put in trace statements
- Try to think like a machine!

Programming Tips

4. Use proper style

- Variable names
- Braces
- Indentation
- Comments
- In case someone looks at your code
- For your own purposes... Write code like your memory will be erased tomorrow!

Programming Tips

3. Learn by experimentation!

- If you're not sure how something works, try it!**
- If you see a technique you're not familiar with try it!**
- You will learn best by thinking about things in different ways**

Programming Tips

2. Programming slowly is faster!

Two kinds of programming:

Preventative: Carefully implement each statement, thinking about what you are doing and considering all possible scenarios

Corrective: Quickly implementing things, planning to later go back and correct problems

Programming Tips

1. Write code **incrementally**.

- Write a tiny piece of code
- Test it thoroughly
- Test it some more
- Test it again
- When it is perfect, move on to the next tiny piece of code