Using Terminal Window Graphics in CS1

David Hovemeyer and David Babcock

York College

of Pennsylvania

CCSCE 2008
Introduction

Terminal Window Graphics

Projects and Labs

Conclusions
The problem

We have many goals for CS1, sometimes conflicting:

- Make it simple
  - *Many students come in with no prior programming experience*
- Make it rigorous
  - Students must master the essential language features, I/O, etc.
- Make it comprehensive
  - Teach the entire language
- Make it fun!
  - Colorful, graphical, interactive
  - A positive experience in CS1 is essential for retaining CS majors
Stuart Reges, *Back to Basics in CS1 and CS2*, SIGCSE 2006

The objects early approach is hard to do right

So, focus on the essentials of procedural programming: types, values, variables, loops, decisions, functions, arrays

- Move on to OOP once the fundamentals have been mastered

Our CS1 course uses C

- Structs + accessor functions = encapsulation
- Easy transition to Java in CS2
How to make the CS1 projects and labs interesting?

- Console I/O: can be boring
  - Oh, another program that reads from stdin and writes to stdout. Whee.

- GUIs/graphics: complex API, takes time away from teaching essentials
  - Wrapper API can help: e.g., acm.graphics package
  - But wouldn’t it be nice if we could somehow do graphics “for free”?
• One of Reges’s CS1 projects: draw a rocket using text characters
• Nice!
  • Interesting visual effect
  • Students learn console output on day 1 of course: *no new API to teach*
• Our thought: why not draw *graphics* in the terminal window?
  • Color, animation, etc.
  • The only requirement is that you can move the cursor!
Flashback to 1983!

- I was in 6th grade
- Family computer: the Kaypro II
- No bitmapped graphics
- Had two (quite good) animated games
  - CatChum (Pac-Man™)
  - Ladder (Donkey Kong™)
- Stephen Ostermiller wrote a free clone of Ladder in Java (ostermiller.org/ladder)
Introduction

Terminal Window Graphics

Projects and Labs

Conclusions
Terminal Window Graphics

- Treat the terminal window as a (low-res) graphics display
- Move the cursor to put characters at arbitrary positions
- Set foreground/background colors
- Update entire screen, delay, repeat: animation
  - Double buffering to eliminate flicker
- Can be portable
  - ncurses on any Unix
  - Win32 console API on Windows
- Open source:
// Output functions
void  cons_clear_screen(void);
void  cons_move_cursor(int  row,  int  col);
void  cons_change_color(int  fg_color,  int  bg_color);
void  cons_printw(const char  *fmt,  ...);
void  cons_update(void);
// Other functions
int cons_get_screen_height(void);
int cons_get_screen_width(void);
int cons_get_keypress(void);
void cons_sleep_ms(int ms);
Experience

- We have found that students grasp the idea of moving the cursor fairly easily
  - They already understand that \n moves the cursor to the beginning of the next line
- Colors also not too difficult
- Even without having previously seen the API, can introduce it in a lab within 20 minutes or so
struct Scene s =
    create_scene(); // create initial scene

while (true) {
    render_scene(s); // draw it in hidden buffer
    cons_update(); // copy buffer to screen
    cons_sleep_ms(Delay); // pause
    s = update_scene(s); // update state of scene
}

Students write the functions in **bold**. Works for any animation, including video games.
Introduction

Terminal Window Graphics

Projects and Labs

Conclusions
Lab: Bouncing Character

- Animate a character moving diagonally, bouncing when it hits the edge of the window
- Use a struct to keep track of position and direction/velocity of the character
Lab: Bouncing Particles

- Animate a large number of characters bouncing around the window
- Use struct to keep track of state of each particle
- Use floating point variables for position and direction/velocity
- Students write accessor functions taking instances of their particle struct type by reference using pointers
  - If you can model one, you can model any number of them!
Assignment: Sudoku

- Write a program to fill in a Sudoku puzzle
- Essentially, a GUI!
- Uses a 2-D array as the underlying data structure

Demo
Assignment: Planet Simulation

- Simulate a planet orbiting a star
- Compute effect of star’s gravity on planet at each time step

Demo
Assignment: Shoot ’Em Up

- Essentially, Space Invaders (tm)
- Use structs to keep track of state of player ship, enemy ships, missiles
  - Manage complexity using accessor functions, encapsulation
- Students write a video game in their first programming course
  - We given them very little skeleton code
  - Hopefully, a confidence-builder

Demo
Assignment: Eat ’Em Up

- Essentially, Pac-Man(tm)
- Just about any 2-D video game could be adapted for terminal window graphics

Demo
Introduction

Terminal Window Graphics

Projects and Labs

Conclusions
Conclusions

- Terminal window graphics is a very lightweight approach to introducing graphics and animation in CS1
- We’ve received positive feedback from students
- Open source: http://libtermgraph.sourceforge.net
  - Links to projects
Questions?