A SUMMER HIGH SCHOOL COMPUTER GAME PROGRAMMING CURRICULUM
AND AN ASSESSMENT OF ITS EFFECTS ON STUDENT MOTIVATION

Phaedra Krizo
B.A., University of California, Santa Barbara, 1995

PROJECT

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

at

CALIFORNIA STATE UNIVERSITY, SACRAMENTO

FALL
2011
A SUMMER HIGH SCHOOL COMPUTER GAME PROGRAMMING CURRICULUM
AND AN ASSESSMENT OF ITS EFFECTS ON STUDENT MOTIVATION

A Project

by

Phaedra Krizo

Approved by:

__________________________________, Committee Chair
V. Scott Gordon, Ph.D.

__________________________________, Second Reader
Mark A. Rodriguez, Ph.D.

__________________________________
Date
Student: Phaedra Krizo

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the Project.

__________________________, Graduate Coordinator       ________________
Nikrouz Faroughi, Ph.D.    Date

Department of Computer Science
Abstract

of

A SUMMER HIGH SCHOOL COMPUTER GAME PROGRAMMING CURRICULUM
AND AN ASSESSMENT OF ITS EFFECTS ON STUDENT MOTIVATION

by

Phaedra Krizo

In the U.S. there are not enough students pursuing STEM degrees to keep up with workforce demand. Additionally, there is a low number of minorities and women in STEM fields. Introducing computer science and other STEM topics in K – 12 classes motivates students to consider related future careers. An ITEST Summer Game Programming Course was initiated at CSU Sacramento to contribute toward addressing these issues. This project has two parts. The first is the creation of a formal curriculum derived from the instructor’s curriculum notes and from instructor and student feedback. The second aspect of the project is an analysis of the effectiveness of the course on STEM interest derived from data from student daily class journals.

This project resulted in a reusable six week game programming curriculum. From analysis of the student journals, it can be concluded that this course was effective in motivating student interest in computer science. The game programming delivery method is fun, interesting and relevant to high school students. It also enabled students to practically apply the computer science and math skills learned, contributing toward retention and creating a meaningful learning experience.

_______________________, Committee Chair
V. Scott Gordon, Ph.D.

____________________________________
Date

iv
ACKNOWLEDGMENTS

Thanks to Devin Cook for allowing the reprinting of the great slides and course materials, to my project advisors for the direction and availability and to my family and my son Will for their general awesomeness.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Acknowledgments</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>vii</td>
</tr>
<tr>
<td>Chapter</td>
<td></td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1. Problem Statement</td>
<td>2</td>
</tr>
<tr>
<td>2. BACKGROUND</td>
<td>6</td>
</tr>
<tr>
<td>2.1. ITEST Goals</td>
<td>6</td>
</tr>
<tr>
<td>2.2. An ITEST High School Game Programming Course</td>
<td>6</td>
</tr>
<tr>
<td>3. PROJECT OBJECTIVES AND METHODOLOGY</td>
<td>9</td>
</tr>
<tr>
<td>3.1. Project Objectives</td>
<td>9</td>
</tr>
<tr>
<td>3.2. Methodology</td>
<td>10</td>
</tr>
<tr>
<td>3.2.1. Curriculum Methodology</td>
<td>10</td>
</tr>
<tr>
<td>3.2.2. Motivation Methodology</td>
<td>11</td>
</tr>
<tr>
<td>4. RESULTS</td>
<td>13</td>
</tr>
<tr>
<td>4.1. Curriculum Results</td>
<td>13</td>
</tr>
<tr>
<td>4.2. Motivation Results</td>
<td>13</td>
</tr>
<tr>
<td>5. CONCLUSIONS AND FUTURE WORK</td>
<td>20</td>
</tr>
<tr>
<td>Appendix A: Summer High School Computer Game Programming Curriculum</td>
<td>22</td>
</tr>
<tr>
<td>Appendix B: Data from Journal Analysis</td>
<td>140</td>
</tr>
<tr>
<td>References</td>
<td>145</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Projected 2018 Computer/Information Specialties Jobs vs. Graduates</td>
<td>2</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Georgetown University Study on Increase in Computer and Mathematical Science Occupations</td>
<td>3</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Projections Data from the U.S. Bureau of Labor Statistics National Employment Matrix</td>
<td>4</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Minorities and Women in Science and Engineering Occupations: 2006</td>
<td>5</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Women as a Percentage of Selected Occupations</td>
<td>5</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Daily Average of Student Skills Self-Perception in Game Programming Course</td>
<td>15</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Daily Average of Student Enjoyment Based on Game Programming Course</td>
<td>16</td>
</tr>
</tbody>
</table>
Chapter 1

INTRODUCTION

By 2018 there will be over five times as many U.S. job openings in science, technology, engineering, and mathematics (STEM) fields than qualified college graduates (Figure 1) [1]. This is due to the rate of growth in technology and other STEM areas, a boom of new retirees and inadequate K – 12 preparation in math and technology. A related issue, and one that contributes to this shortage, is the lack of ethnic and gender diversity in the field.

In order to attract college students to major in STEM fields such as computer science, introduction to these areas needs to occur before students begin post-secondary education. One method of motivating students to pursue a STEM degree is to increase implementation of programs at the K – 12 level with the objectives of exposing students to basic topics and concepts, teaching principles and skills and sparking student interest in the subject.

An Innovative Technology Experiences for Students and Teachers (ITEST) High School Game Programming Course was created as a partnership between California State University Sacramento (CSUS) Department of Computer Science and College of Education, Mathematics Engineering Science Achievement (MESA) and local K - 12 schools to provide educationally disadvantaged students with the opportunity to learn basic computer science skills and to provide encouragement for post-secondary education in a related field. This project presents a formal curriculum for the course and an analysis of its effectiveness on student motivation.
1.1 Problem Statement

In the 19th and 20th centuries, electricity and the internal combustion engine drove the rise of manufacturing and America’s shift away from an agrarian economy. Today, computers and related inventions are driving the information revolution and transforming the U.S. economic landscape once again. And, just as the industrial revolution was critical to building a mass K–12 education system to feed workers into the manufacturing industries, the information revolution is spurring the development of a mass postsecondary system to fill the needs of sophisticated new industries, such as computer systems design... [2].

Software Engineer is the top job in the U.S. for 2011, with Computer Systems Analyst at in the fifth position, according to a study published in Forbes based on U.S. Bureau of Labor statistics, using the criteria of pay, outlook, work environment, stress and physical demands [3]. By 2018 the number of Computer and Mathematics related jobs in the
U.S., the largest group of STEM occupations, is projected to reach 4.2 million (Figure 2) [2]. Total jobs for Computer Software Engineers and Computer Programmers alone is projected to surpass 1.6 million with an increase of over 20% from 2008 (Figure 3) [4]. Why aren’t high school graduates clamoring to prepare for a field that not only has many jobs, but has many of the nation’s top jobs? Part of the problem is lack of K - 12 preparedness, especially in mathematics, for pursuing a STEM major, and negative perceptions about or unfamiliarity with computer science related professions and job opportunities.

Figure 2: Georgetown University Study on Increase in Computer and Mathematical Science Occupations [2]
Another facet of the problem is the low number of minorities and women in STEM fields in the U.S. In 2006, women and minorities combined comprised 45% the current U.S. STEM workforce (Figure 4) [5]. In 2009, women held around 25% of Mathematical and Computer Science positions, despite comprising about half of the workforce (Figure 5) [5]. Addressing this issue also contributes to reducing the overall shortage in STEM fields.
Figure 4: Minorities and Women in Science and Engineering Occupations: 2006 [5]

Figure 5: Women as a Percentage of Selected Occupations [5]
Chapter 2

BACKGROUND

The ITEST High School game programming class initiated at CSU Sacramento was created in response to the above issues. The ITEST project proposal describes one of the outcome goals as follows:

*Being involved in the program will boost confidence of the participating underrepresented student groups, demonstrate to them how science is relevant to their world, and catch girls in the middle school years before they lose interest, thereby positively affecting their choices of math and science courses in high school* [6].

2.1 ITEST Goals

The goal of ITEST, a National Science Foundation (NSF) program, is to address the current and projected shortages of professionals in STEM fields through enhanced STEM education for K – 8 students. The three types of ITEST programs are *scale-up*, which are implemented at broad regional or national levels, *strategies*, which target students and teachers, and *research* [7]. The program discussed here falls under Strategies. “The strategies are intended to encourage students' readiness for, and their interest and participation in, the STEM and ICT [information communications technologies]-intensive workforce of the future” [7].

2.2 An ITEST High School Game Programming Course

The complete ITEST High School Game Programming course began in Summer 2011 and concludes at the end of Fall 2012 for a first group of forty students, and runs
from Summer 2012 through Fall 2013 for a second. The primary focus of the program is to provide STEM opportunities to underrepresented students. Both two-week summer sessions are taught on the university campus by a CSUS Computer Science instructor and a high school teacher from one of the participating Sacramento area schools. The Fall and Spring semesters are taught as part of the students’ curriculum by a high school teacher.

The curriculum teaches computer science and math skills using a game programming format. The first session used the TeachScheme functional educational programming language. Scratch, a visual educational programming language, will be used in the second session. By the end of the first six week Summer tracks the students create a simple computer game. By the end of the entire program, the students will produce a more elaborate math skills game geared toward kindergarten age children. The course also includes a number of workshops which teach educational and life skills and feature speakers who provide encouragement and inspiration to pursue post-secondary education. Additionally, using a mutual mentoring environment model that spans the entire ITEST target audience, middle school aged students will be involved in the game concept planning phase and will facilitate game testing by elementary school students. “[Studies] have shown that a purely curricular approach is insufficient for reaching underrepresented groups, and that mentoring can lead to better career development as well as reducing the likelihood of students dropping out of high school” [6].

Following are goals outlined in the course proposal.

1. **Students will be more motivated to pursue STEM fields in college.**

2. **Students will be better prepared for STEM studies in college.**
3. An effective curriculum will be produced that can be used not only at other schools with underrepresented students, but also to college freshmen and high school students in general [6].

The ITEST program produces a curriculum and research. The research focuses on the effect of the program on student motivation, plans, STEM interest and preparedness and computer science skills learned.
Chapter 3

PROJECT OBJECTIVES AND METHODOLOGY

3.1 Project Objectives

This project focuses exclusively on the Summer 2011 session of the CSUS High School Game Programming course. There are two facets to the project. Both relate to the ITEST strategies project type, which focus on designing, implementing and evaluating student learning experiences with the goal of encouraging student STEM interest and readiness for a future related career [7].

The first aspect of the project is the conversion of the student and instructor journal entries into a reusable curriculum in a daily lesson plan format. The formal curriculum is derived from the instructor’s curriculum outline and notes, the student work and student feedback on the activities. The curriculum also includes comments based on feedback from the daily student and instructor journals, for example regarding the difficulty or effectiveness of a particular activity.

The second phase produces analyzable data derived from the student journals. The daily journals will be assessed based on two areas: enjoyment of the day’s activities and of the class overall and self-perception of increase in computer science skills. The data is graphed to identify and illustrate trends throughout the summer program and to provide an in-progress assessment of its effects on STEM interest in participating students.
3.2 Methodology

This section describes the methods used to execute each section of the project, including a description of original data used, data extraction methods, instruments and methods of analysis, and, for the curriculum section, curriculum format.

3.2.1 Curriculum Methodology

A variety of lesson plan formats were reviewed for appropriateness for the subject area, material and age group. The final version was inspired primarily by four sources: K-12 lesson plans on the NASA Teachers’ Corner website [8], Colorado State University’s Writing Lesson Plans Teaching Guide [9], the CyberCampus “Tips for writing performance-based objectives” [10] which is based in part on principles from Bloom’s Taxonomy, and the Bootstrap Curriculum format [11]. Additionally, many of the activities in the course were based on Bootstrap [11] and Picturing Programs [12] curriculum and the program reuses some of the materials, such as workbooks and demo material. The lesson plans were compiled from notes on the daily lesson plans by the CSUS instructor, Devin Cook, from the course materials, including PowerPoint slides, worksheets and assignment sheets, and from the student work.

The Reflection section of each daily lesson plan describes what was and was not effective in the first implementation of the curriculum. Material was extracted from the instructors’ journals and from feedback in the student journals on each day’s activities, if relevant. For example student feedback would be used if a majority of the students did not understand a particular topic or lesson or expressed that an activity was a success.
Student feedback on the workshops and special events is also included in the Reflection section.

3.2.2 Motivation Methodology

Analysis of student motivation was drawn from student journals. There were forty students in the class, and journals were recorded in on twenty seven of the class days; in all, over a thousand journal entries were read and assessed. The format is freeform, except for the period from days 11 – 19 where a series of questions were posed to the students to respond to in their journals: “Did you enjoy today’s class and material,” “How difficult/easy was the material today” and “What do you think about your computer science skills overall?” As the students tended to write a little more in the freeform format and responded more positively to that format of the activity, the questions were discontinued. The results from both journal styles were similar enough that it did not affect the analysis.

Because of the freeform nature of the journals, the data extracted was sometimes inferred by the students’ choice of words, so is not exact. Each of the journals was analyzed by class day for comments pertaining to two areas: enjoyment of the class and activities and skills perception (perception of skills learned that class day or advancement of overall computer science skill level). If references were found, the student’s journal day was assigned a numerical value for each area. The levels of enjoyment are: 5 – Very Fun, 4 – Mostly Fun, 3 – OK, 2 – Mostly Boring, 1 – Very Boring. Skills perception is
rated: 3 – Great Improvement, 2 – Medium Improvement, 1 – Little Improvement.

Journal content was also assessed semantically for motivation and STEM perception.
Chapter 4

RESULTS

4.1 Curriculum Results

(See Appendix A for complete curriculum).

The curriculum employs the following layout for each class day: time frame, overview, learning objectives, assessments, materials, instructional procedures, closure, reflection and reference. Time frame indicates length of the class day. Overview is a summary of the day’s activities. Learning objectives are what the students will be able to do after the day’s lesson, for example compare strings using ASCII values or use key handlers to move characters on the screen. Assessments are tools, such as quizzes, that gauge student progress or comprehension. The instructional procedures section details the day’s lessons and activities. Closure specifies any end of day actions such as turning in assignments or journal time. The reflection section draws from the student journals and course instructor Devin Cook’s journals, noting when there are pertinent comments on the day’s curriculum or material.

4.2 Motivation Results

Attracting students to STEM courses is only part of the challenge. Once students arrive, they must find qualities that entice them to stay and pursue the subject further. A class, especially when it is the student’s introduction to the material, must be exciting, fun or present interesting challenges. Ideally it will have all of these qualities.
One study on high school students’ perceptions of computer science offers the following description of curriculum goals.

The goal of [high school computer science] curriculum is to prepare and motivate [students] for careers in today’s expanding, Internet-based, global economy. We suggest that by portraying computing as an innovative, creative, and challenging field with authentic, real-world applications, we may be able to motivate teenagers to become more excited to pursue careers in computing [13].

The game-based content delivery proved to be extremely relevant and motivating for this age group. Benefits include providing a familiarity with computer science topics and relevancy, counteracting commonly held negative stereotypes about computer related jobs and, crucially, providing an exciting, rewarding experience. Studies on similar programs report the same results. “…research has suggested that a “games first” approach in a CS1 course resulted in higher retention and had a positive influence on both women and men, without any loss in the technical skills” [13].

Producing a final, tangible (and fun) product rather than focusing on individual skills makes the experience more meaningful. Instead of learning skills in isolation, students can see immediately how learning how a particular skill or concept can be applied practically when they create their games. Students won’t have to wonder, for example, where outside of this classroom could it possibly be useful to know how to convert infix to prefix notation? This results-based approach succeeded in boosting student confidence in their computer science skills. The average daily rating for self-perception of increase in skills, based on the journal analysis, is 2.7, between Great Improvement and Medium Improvement (Figure 6). (See Appendix B for data tables by class day).
This method of content delivery also helped to counteract some of the stereotypes studies have shown students to have regarding computing professions.

*We found that teenagers perceived computing to be boring, solitary, and lacking real-world context, yet graduate students described their research as exciting, social, and having a direct and meaningful impact on the world around them.... Students do not see computing as an environment for creativity and believe that it focuses on tedious details instead of important topics in problem solving. Furthermore, students from underrepresented groups often perceive themselves to be unwelcome in the field, feel uncomfortable asking for help, and find the culture surrounding CS to be aggressive [13].*

This class succeeded in illustrating that computer science isn’t boring or unwelcoming.

The average overall daily rating of enjoyment for the course is 4.2, between Mostly Fun and Very Fun (Figure 7). Working in teams to create games creates a cooperative, social, inclusive environment and demonstrates that programming isn’t a lonely, solitary experience. The experience of designing and creating their own games provides students

---

1 The wave pattern on the Student Enjoyment graph generally corresponds with the school week. Low points are on Mondays when students realize that they are going to school in the summer and are often tired after the weekend, climb steadily throughout the week and peak on Fridays, which are enrichment activity days.
with confidence that they have the ability to program, that it isn’t too difficult or boring.

Following are a few excerpts from the daily journals.

“To tell the truth, at first I really thought that computer science was really boring. But then now, even though it’s only my second day here, I’m learning more than I need to know already! It’s not boring after all.”

“Today we finished up our game; the last step was to have the score come up in our game. Overall looking back it wasn’t as difficult as I thought. I was frustrated at times but it was worth it, the outcome looks nice.”

“I really love computer science.”

“This class is really interesting and at first I thought that it was going to be a boring, nerdy class. However, this class is so awesome! I absolutely love it! Again, I am very happy that I am getting to be a part of this program. I want to learn a lot more and am ecstatic to expand my knowledge in computer science. I never really knew exactly what it was, but now that I am getting the hang of it, I believe more people should look into it, they might find it really fascinating.”

Figure 7: Daily Average of Student Enjoyment Based in Game Programming Course

However, some stereotypes persist. One student expressed that she was undecided about entering a field such as Computer Science that she perceived as male-oriented, even though she enjoyed it immensely.
“I really love designing games. I think that it’s the awesomiest thing! Somehow I want to become a game designer now. But then on the other hand, I feel like [people are] going to make fun of me, saying that I’m not even a guy so why am I doing this. But on the other, other hand I feel like whatever! This is freedom so I can do whatever I want even though I’m not a guy.”

Hopefully this perception will change by the end of the program. This outlook is a good indication that stereotypes such as “math is for boys” need to be addressed at the grade school age group or younger.

Creating their own video games added a level of excitement. Students often commented in their journals that they couldn’t wait to make their characters move, or to see what they would be doing the next day or to be able to play their completed games. There was also a great deal of pride expressed regarding their creations. The following are four students’ journal entries.

“Not a lot of high school students get to say that they created a video game. I feel very lucky!”

“Today was pretty RAD....Well, after lunch we worked on animations. It’s pretty cool how we can make the images move any way we want. I’m getting so pumped to finally see action. Thanks, NSF.”

“Today went well. We finish our game, now it’s all moving and its very fun! I’m so happy right now. Why? Because I just finished my first game! Omg I did it!! And now I can’t wait to show my grandma and my big brother my first game I made.”

“Today was a really fun day. We finally finished our video game! I’m so excited to show my mom what I created! I am really happy I accepted to be a part of this program! I can’t wait to return next summer!!!!!!!”

Another benefit of this course is offering encouragement for pursuing higher education. Just by attending a college program while in high school, students have an increased likelihood of completing high school and of attending college. Also, students
were positively influenced by the workshops. Several workshops, lectures and special events were held during the summer course focusing on practical and educational skills, such as money management and college essay composition, and on providing motivation or inspiration to pursue STEM fields and higher education in general, including motivational speakers and a college tour. Students responded well to all of the workshops. Particularly inspiring to students is a motivational lecture by a college graduate. She discussed her background and how she got her scholarships. Many of the students identified with her as coming from a family where no one has attended college. Following is one student’s response to the lecture.

“Her story inspires me to go to college and become someone in this world. I really can’t wait to go to college because I will be the first in my family... The college seminar today was really interesting. I usually have only had a person who is still a college undergraduate speak to me. So to have someone speak to us who has already graduated from college with a double major and a triple minor, was extremely helpful and interesting. I really enjoyed listening to her story and her background. It made it even better that she was not a snotty rich kid, but someone who came from a huge family that grew up in poverty with a bad mentality. Which is what is usual for students like us. It is always fascinating and comforting to hear about journeys from “underdogs,” so to speak. I will be attending more college seminars for sure!”

Many of the students expressed gratitude toward the program and teachers and felt fortunate to be a participant. The experience and activities were regarded positively and many students expressed excitement in their journals about continuing through the school year and the next summer and about creating another game. Following are four students’ end of summer reflections.

“I really can see myself creating games as a life career. I love creating things of my own. If you love what you do, you will never have to work a day in your life. I really think this could be my major in college.”
“I’m going to miss this program! ! ! ! !”

“I really like this program and am happy that I was selected to be a part of it... I’m excited for the computer science class to see what new things we get to learn. I am really excited to start playing our finished video game!”

“Math is awesome. I love math.”
Chapter 5
CONCLUSIONS AND FUTURE WORK

From the analysis of student journals, it can be concluded that this game programming course is effective in encouraging student interest in the STEM field of Computer Science, and in higher education in general. It provides excitement, fun and of course computer science education which inspires in students motivation to further pursue the subject. The class also assists in counteracting prevalent stereotypes about computer programming, for example that it’s a boring, solitary activity and is not for girls. Even if a student doesn’t choose to pursue a STEM field, by participating in this type of program s/he already is benefiting from a lowered chance of dropping out of high school and an increased chance of pursuing higher education.

On a broad scale, there is vast potential for future work in the area of K-12 STEM education. Considering how far behind K-12 STEM education currently is to where it needs to be to meet workforce demands, and considering the rate of change in technology that creates a perpetual need for up-to-date courses and curriculum, there is much work to be done in this area. Increased concentration on K – 12 computer science education and other STEM programs is important. Targeting early grade school to prevent or counteract the stereotype that math and science is for boys only would open up the field to women who might not otherwise have considered it as an option for the future. Also important are programs that focus on presenting a more appealing image of STEM, especially to underrepresented students, such as President Obama’s Educate to Innovate campaign that promotes STEM fields through a partnership with the private sector [14].
On a micro level, there are opportunities for future work related to the program discussed here. At its conclusion a study of the effectiveness of each programming language, TeachScheme and Scratch, would address the third goal of the program: “To assess the efficacy of two computer languages designed for K – 12 students” [6]. Extended future work could include a long range study to follow-up with the two groups of students to determine the percentage that attend college and pursue STEM majors compared to the national average, based on demographics. A shorter range study could follow-up with students within months after program completion to study the persistence of its effects on future goals. Another area for research is an assessment of skills learned. The implementation of pre and post testing instruments and an analysis of the course compared to a lower division college course could be conducted. For example, at CSUS would the program meet requirements for a Computer Science course equivalent? Other potential future work is the implementation of surveys at the end and beginning of the course with the purpose of finding out the effectiveness of the course in areas such as perception of STEM fields, i.e., “What is computer science?” and “What does a computer scientist do?”, perceptions of minorities and women in technical or STEM fields, plans to attend college after high school, plans to major in a STEM field if pursuing post-secondary education and the likelihood the student will pursue a career related to STEM.
APPENDIX A

Summer High School Computer Game Programming Curriculum
Summer High School Computer Game Programming Curriculum

Age group: High School

Program length: 6 weeks

Subject: Computer Science/ Game Programming

Requirements:
Computer lab with DrRacket programming environment installed on each system.

Overview:
This course is an introduction to computer science and programming skills. Students will create a basic computer game by the end of the six weeks using the TeachScheme Racket programming language.

Topics:
Topics covered include: functions, variables, input/output, strings, computer coordinate system, data types, Boolean logic, logical operators AND, OR and NOT, shape and image manipulation, simple animation, tick and draw functions, code conventions and formatting, if and conditional programming structures, decision making, mouse, key and event handlers, character placement, game worlds, animation speed, collision handling and program termination.
PROGRAM DAY 1 – WEEK 1: DAY 1

Time Frame: 1 six-hour class day

Overview: This is the first day of class and many of the students’ first time in a college environment. The day will be spent on welcome, introductions, rules, setting behavior expectations and campus orientation.

Learning Objectives: Students will be able to:
- Distinguish appropriate general behavior and behavior towards others in a college environment
- Paraphrase lecture and lab rules and apply them in the classroom

Assessments: In class observation

Materials: Slides 1.1 – 1.5

INSTRUCTIONAL SEQUENCE

Instructional Procedures:

Introduction
- (Slide 1.1)
- Welcome Message
- Teacher Introduction

Class Expectations
- You are college students (Slide 1.2)
- Lecture vs. Discussion
- Rules (Slides 1.3 – 1.5)
  - In-class Rules
  - Lab Rules
  - General Rules
Reflection:
More content delivery was originally planned for this day than was feasible. In the future this day will be dedicated solely to orientation as there didn’t end up being enough time to begin lessons.
Welcome Everyone!

Slide 1.1

Welcome to Sac State

- Welcome to Sac State!
- Since you are in a college
- ... and you are students
- ... consider yourselves college students!
- But, how do you "act" like college students?

Slide 1.2
In-Class Behavior

- No electronic devices!
  - Phones
  - Laptops
  - Games
  - etc...
- Don't talk to other students
- No sunglasses
- No eating

Slide 1.3

In-Lab Behavior

- I will often put you into groups – these will change often
- Labs include programming challenges and activities
- When I tell you to turn your monitors off – you must do so immediately!
- **NO** browsing the Internet. **NO** Facebook. **EVER!**
Other Rules

- Never insult another student – call them dumb, taunt, etc...
- I take a very dim view of this
PROGRAM DAY 2 – WEEK 1: DAY 2

Time Frame: 1 six-hour class day

Overview: Students are introduced to course material and video game basics. Backwards engineering, (computer) coordinate planes, defining programming languages, values, an introduction to Racket and infix notation will be covered. Students will brainstorm for game ideas.

Learning Objectives:
Students will be able to:
• Use backwards engineering to dissect basic components of a simple computer game
• Determine coordinates of an object on a screen and describe what happens to the coordinates when an object moves
• Articulate what an algorithm is and provide examples
• Give examples of a value
• Use Racket to get results for simple function

Assessments:
• In class observation
• TURN-IN

Materials:
• NinjaCat demo
• Worksheets 2.1
• Bootstrap Workbook pp. 1,3
• Slides 2.1, 2.2
• Paper Cutouts (Bootstrap Workbook p.2)
• Student Glossary (2 pages)
• Colored pens

INSTRUCTIONAL SEQUENCE

Instructional Procedures:
Backwards Engineering
• QUESTION: What is backwards engineering?
• DEMO: NinjaCat
  o NinjaCat is a game made using the same tools that will be used to create our games in class
• GROUP ACTIVITY: Objects in Demo (15 min.)
  o (Bootstrap Workbook p.1) We’ll use the table to reverse-engineer NinjaCat to see how it works.
    ▪ Who can name something they saw in NinjaCat?
In groups, make a list of 15 things from the demo

Coordinate Planes
- QUESTION: How do we describe the location of each object in a game?
- ACTIVITY: Paper Cutouts (Paper Cutouts)
  - Spend time placing paper cutouts of NinjaCat, Ruby and Dog on a grid on the whiteboard to illustrate coordinate system
  - What happens on the coordinate system when objects move?
  - (Slide 2.1) What is NinjaCat’s coordinates? What would his coordinates be if he was in the top left corner, etc.
  - QUESTION: What would a character’s coordinates have to be to be off the screen?

More Dissection
- PAIR ACTIVITY: (Bootstrap Workbook p.1, Column 2) What is changing?
  - Ask about each object: ruby, cat, dog, clouds
  - Computer coordinate system – 0 is at the top

Video Games: Make a Game
- Video Game Basics
  - Most action video games consist of a player that must obtain a goal
  - The goal is usually touched – whether it is an item on screen or a finish line. To make the game fun, something interferes with the player – the "danger".
  - (Slide 2.3) Example: Basic Atari 2600 games
  - Basics of game we will create: each will have player, goal, danger
  - Form new teams of two
  - PAIR ACTIVITY: (30 min.)
    - (Bootstrap Workbook p.3) Design game
    - (Worksheet 2.1, colored pens) Sketch it
    - Both team members fill out worksheet and draw

Glossary
- (Student Glossary) Handout glossary sheets for writing down vocabulary terms throughout the course

What is a Programming Language?
- Language – like Hindi, Spanish
- Number of languages today
- VOCABULARY: whitespace, syntax, semantics, algorithms
- Algorithms
- QUESTION: What are some examples of algorithms?
Circles
- Draw a circle on the white board
  - The circle is our computer
  - Put numbers in it, get results
  - One operator, many values
- Student open Racket
  - Point out Definitions and Interactions panes
  - Simple numbers: enter a number, a number is returned. This is an example of values.

Infix notation
- Back to Circles
- Work up to expressions with multiplication – many levels deep
- Show students how to do it in Racket
- Explain that the (...) is a "function"
- Give infix, have student and partner come up with result

Closure:
- What did we learn?
- Who saw someone else in the class do something great?
- What was the hardest part so far?
- TURN IN: Worksheet 2.1 (with names on it)
- Daily Journal

Reflection:
  Colored pens were added to the curriculum to make the game object drawings more interesting.
  Video game basics was added to the curriculum. As an example, an Atari 2600 emulator was used to show some basic, primitive games. This is an interesting demonstration to include – the students responded well to it and, surprisingly, were familiar with the games and knew what an Atari 2600 is.

Reference:
  Bootstrap Curriculum, Unit 1 [11]
Your First Game

Names: _____________________________________________

Game Name: _____________________________________________

Drawing for the: □ Player       □ Goal       □ Danger       □ Background

Worksheet 2.1 (Created by Devin Cook)
Show picture of Ninja Cat from Demo

Slide 2.1

Show screenshot of PacMan

Slide 2.2
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Student Glossary - Page 2  (Created by Devin Cook)
PROGRAM DAY 3 – WEEK 1: DAY 3

Time Frame: 1 six-hour class day

Overview:
Coordinate system, infix notation and parenthesized expressions are reviewed. Contracts and Strings are introduced.

Learning Objectives:
Students will be able to:
• Convert infix expressions into fully parenthesized expressions
• Name the components of a contract and define a contract and each component
• Articulate definition of input and output and give examples
• Use simple String functions

Assessments:
• In class observation

Materials:
• Slides 3.1 – 3.7
• Bootstrap Workbook pp. i,5

INSTRUCTIONAL SEQUENCE

Instructional Procedures:

Introduction/Review
• ACTIVITY: Red Ball in the Lecture Hall (Coordinate planes review)
  o A red vinyl ball is used to represent the ruby from the NinjaCat game. In the lecture hall (arranged in a grid pattern) students pass the ruby around. To get the ruby each pass it by the difference in x and y coordinates. The idea is to underscore that, in games, objects move relative to their current position.
• (Slides 3.1 – 3.3) Infix notation
  o Slides demonstrate how to turn any infix expression into a fully-parenthesized expression. Conversion to prefix simply requires the operator to be moved.

Fully Parenthesized Expressions
• (Slides 3.4 – 3.7)
Contracts
- **QUESTION:** What is a blueprint?
- **CD example:** Point out that what's on the CD does not matter
  
  \[
  \text{CD Player : CD} \rightarrow \text{Music} \\
  \text{; Plays the music from the CD}
  \]

- **QUESTION:** What is a blueprint for a coffeemaker? How about a light bulb?
- **ACTIVITY:** Have students think of several more examples
- **Input / Output**
- **Domain** – Put coffee in a CD player?
- **Range** – Coffee maker can make pizza?
- **Racket is the same.**
  - Math requires 2 (or more) numbers
  - They can be any numbers
- **Blueprint \rightarrow "contract"**
- **It is a promise – you give me this, I will give you that**
- **We need to write down the contracts**
- **(Bootstrap Workbook p. 1)**
  - Name
  - Domain
  - Range

In-Lab Practice
- **GROUP ACTIVITY:** (Bootstrap Workbook p. 5) Circle Competition (20 min.)
- **Challenge problems**

Strings
- **Example:** What doesn't fit: 6, "cat", 0, -2, 7.5
- **Two forms:** numbers, text
- **Text is known as a string**
- **Double quotes**
- **Math functions don't work on strings – duh!**
- **We have string functions – one is called string-length**
  
  \[
  \text{; string-length: String} \rightarrow \text{Number}
  \]

- **QUESTION:** What is the domain, range
- **QUESTION:** What do you think it does?
- **INDIVIDUAL ACTIVITY:** Find out what it does!
  
  \[
  \text{; string-append: String, ...} \rightarrow \text{String}
  \]
Closure:
• Daily Journal

Reflection:
At this point the vocabulary got unwieldy. (The student glossary list was added to Curriculum Day 2).

Reference:
Bootstrap Curriculum, Unit 2 [11]
Dang, these expressions are weird!

Slide 3.1

**Infix Notation**

- Using *infix notation*, we put the operating in between the two operators
- This is the standard format used today

<table>
<thead>
<tr>
<th>To add the numbers ( a ) and ( b ), we type:</th>
<th>( (a + b) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>To divide ( a ) by ( b ), we type:</td>
<td>( (a / b) )</td>
</tr>
</tbody>
</table>

Slide 3.2
Prefix Notation

- *Prefix notation*, rather than putting the operator between the operands, puts it first
- It is also called "Polish Notation"
- This notation is used by the Racket programming language

To add the numbers \( a \) and \( b \), we type: \((a b)\)

To divide \( a \) by \( b \), we type: \((a b)\)

Where are My Parenthesis?

<table>
<thead>
<tr>
<th>Infix</th>
<th>Racket</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1 + 2 \times 3)</td>
<td>( (+ 1 (* 2 3) ))</td>
</tr>
<tr>
<td>( (1 - 2) \times 3)</td>
<td>( (* (- 1 2) 3))</td>
</tr>
<tr>
<td>(1 \div (2 - 3) + 4)</td>
<td>( (+ (/ 1 (- 2 3) 4))</td>
</tr>
<tr>
<td>(1 + 2 \div (3 - 4))</td>
<td>( (+ 1 (/ 2 (- 3 4)))</td>
</tr>
</tbody>
</table>
Converting to Infix to Prefix

- Why are learning this... *be patient!*
- Anywhooo... converting from infix to postfix or prefix notation is easy to do by hand
- Did you notice that the operands did not change order? They were always 1, 2, 3...
- We *just* need to rearrange the operators

1. Make it a *fully parenthesized expression* (FPE) - one pair of parentheses enclosing each operator and its two operands
2. Move the operators to the start of each sub-expression
Slide 3.7

Infix to Prefix

1. \( \frac{1}{(2 - 3)} + 4 \)

2. \( (+ \left( \frac{1}{(2 - 3)} \right) + 4) \)
PROGRAM DAY 4 – WEEK 1: DAY 4

Time Frame: 1 six-hour class day

Overview:
Time will be spend filling Student Glossaries with vocabulary. Creating shapes in Racket and shape contracts will be covered. Variables and functions, and how to create them in Racket, will be introduced.

Learning Objectives:
Students will be able to:
• Define vocabulary words used in class
• Draw shapes in Racket
• Define variables
• Create a function

Assessments:
• In class observation
• TURN-IN

Materials:
• Student Glossary Page

INSTRUCTIONAL SEQUENCE

Instructional Procedures:

Introduction/Review
• Fully parenthesized expressions
• Contracts
  o Input / Domain (and arguments)
  o Output / range
  o Discuss: string-length vs. string-append
• Circle diagrams

Glossary
• (Student Glossary Page) - Handout sheets
• Ask for words we have learned so far
• Have students write up a definition
• VOCABULARY:
  1. Glossary
  2. Coordinate Plane
  3. Programming language
4. Syntax
5. Semantics
6. Function
7. Input
8. Output
9. Domain
10. Range
11. Argument
12. Infix
13. Prefix
14. Fully parenthesized expression
15. String
16. Data type
17. Character

Images
- (require ...), save to Gaia folder
- Draw circle: (circle 100 "solid" "red")
- QUESTION: What is the blueprint / contract?

```
; circle: Number String String -> ...
(circle 100 "solid" "red")
```

- QUESTION: What is the domain?
- QUESTION: What is the range? Try it out!
- It gives back an "image"
- GROUP ACTIVITY:
  - Write down before you test
  - Use it

```
; rectangle: Number Number String String -> Image
```

- GROUP ACTIVITY:
  - Write down each first
  - Create each

```
; ellipse: Number Number String String -> Image
; triangle: Number String String -> Image
; star: Number String String -> Image
; radial-star: Number Number String String -> Image
; regular-polygon: Number Number String String -> Image
```
Defining Variables

- Remind students of Interactions and Definitions panes
- QUESTION: Why is Interactions cleared each time I hit run?
- ANSWER: Interactions is for talking to Racket
- (define ....)
  - Don’t have them write the contract
  - Tell them this is very, very useful
- ACTIVITY: First Variable (10 min.)
  - Have students create a definition for their name
  - Call the variable “name”
  - TELL THEM TO HIT RUN!

```
(define my-name "Mr. Poo")
```

- ACTIVITY: My name is... (10 min.)
  - Have students use string-append in INTERACTIONS
  - Print "My name is ..." using their name variable
  - Show example of string-append with variable usage

- ACTIVITY: My favorite food (10 min.)
  - Have students define their favorite food
  - You don’t have to remove your name define
  - Print to the screen "My favorite food is ...." using string-cat

Defining Functions

- VOCABULARY: whitespace
- Shape examples

```
(define (tri) (triangle 50 "solid" "green"))
```

- Show what happens with gt and (gt)
- Now, move into their own variables

```
(define (tri size) (triangle size "solid" "green"))
```

- ACTIVITY: Modify the contract so you can specify the color
- Don’t record this!
- This is OUR function, not a built-in one!

```
; tri : Number String -> Image
```
Closure:
- TURN-IN: Write a simple function
- Daily Journal

Reflection:
Much of the first hour was spent reviewing fully parenthesized expressions. Students need to understand this concept if they have any hopes of writing complex expressions in Racket. The concept of "contracts" was also reviewed.

After demonstrating circle function many students discovered how to create triangles, squares, stars, and a few more on their own. Function prototypes were not covered, but, given Racket's consistent structure, the students were able to discover them on their own. So, the Racket (LISP) grammar had a definite benefit. Students were given extra time to explore as they were learning content on their own.

In the afternoon, variables and functions were introduced. As this is a major topic in programming, and one that students often have trouble with, the material will be introduced very slowly to ensure all students understand before moving on to more difficult topics.

Reference:
Bootstrap Curriculum, Units 2 - 3 [11]
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Student Glossary (Created by Devin Cook)
PROGRAM DAY 5 – WEEK 1: DAY 5

Time Frame: 1 six-hour class day

Overview: An enrichment activity, Finance 101, is held in the morning. The afternoon consists mostly of working on drawings for the games. The major concepts and terms from the week are reviewed and game skeleton variables are introduced.

Learning Objectives:
Students will be able to:
• Create images for use in their games
• Use a spreadsheet for money management (enrichment activity)

Assessments:
• In class observation
• TURN-IN

Materials:
• Worksheet 5.1
• Colored pens

INSTRUCTIONAL SEQUENCE

Instructional Procedures:

WORKSHOP: Finance 101

Review
• QUESTION: What did we learn yesterday?
• QUESTION: Vocabulary Review
  o Syntax
  o Semantics
  o Function
  o Input
  o Domain
  o Range
  o Argument
  o Infix
  o Fully parenthesized expression
  o Data type
• **ACTIVITY: Hello!**
  o Students will create a function with the contract below
  o It will display a greeting to the screen on one line
  o Students will use string-append in the define

```racket
; hello String -> String

; Ex: (hello "Poo") --> Hello Poo, how are you?
```

**Game Skeleton**
- Download "Game.rkt" from Bootstrap
- Program is inert
- Have students run it
- **QUESTION:** What happens when you type TITLE
- **ACTIVITY:** Change the title of your title
- **QUESTION:** Title Color – What is the name of the next variable defined in this file (TITLE-COLOR)?

**Game Drawings**
- **Demo:** How to use Google images and Paint.net to search for and modify .png images
- **ACTIVITY:** Drawing Your Game (45 min. +)
  o (Resumed activity from Day 2)
  o You can draw images to be scanned or find images online
  o If you draw - keep in the lines
  o Outlined figures works best (Draw examples on board)
  o Have fun!

**Closure:**
- **TURN-IN:** Write a simple function to compute the area of a circle
- **Daily Journal**

**Reflection:**
Students were very interested in the finance presentation and in learning how to use MS Excel for money management.

**Reference:**
Bootstrap Curriculum, Unit 3 [11]
Your First Game

Names:

Game Name:

<table>
<thead>
<tr>
<th>Drawing for the:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Player</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Goal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Danger</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Background</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Worksheet 5.1 (Created by Devin Cook)
PROGRAM DAY 6 – WEEK 2: DAY 1

Time Frame: 1 six-hour class day

Overview:
As most activities are group or pair activities, a quiz is given today to ensure that all students understand the material. As a lot of material was covered last week, most the morning is spent on review. Students will create game skeletons and learn how to place images into the game, including a background and the scanned character images created last week.

Learning Objectives:
Students will be able to:
• Create a game skeleton
• Use place-image and Z-Order to place background and player images into their games

Assessments:
• Pop quiz
• In class observation

Materials
• Download “Game.rkt” from Bootstrap website and install on student machines
• Quiz

INSTRUCTIONAL SEQUENCE

Instructional Procedures:

Pop Quiz
• Scatter students so no one sits together
• Talk about the significance of F’s in a college course
• Bring anyone without a perfect score up to speed before lunch

Review
• Review variables
• Review functions

; hello String -> String
;
; Ex: (hello "Poo") --> Hello Poo, how are you?
Game Skeleton
- Have students run it
- QUESTION: What happens when you type TITLE?
- ACTIVITY: Change the title to the title of your game
- QUESTION: How would we change the title color?

Game Graphics
- Games are screenshots
- Graphics are images placed on images
- QUESTION: What goes on top?
- QUESTION: What goes on bottom?
- This is Z-Order

Game.rkt
- QUESTION: What is BACKGROUND?
- Define a new variable called "SCREENSHOT",
- Set it to BACKGROUND: (define SCREENSHOT BACKGROUND)

Place Image
- Place PLAYER onto the SCREENSHOT

(place-image image x y image)

- PAIR ACTIVITY: Figure out the function

Bitmap
- Get students their scanned graphics
- “bitmap” allows an image to be created from a file

(bitmap String)

- Have them rename files – to something more logical
- Put player on background
- Put danger on background
- Put goal on background
Closure:
• Daily Journal

Reflection:
Be sure to install the library for the Bootstrap skeleton before class – it has to be manually added to the application's folder on each machine.

During the first implementation of the curriculum all students were required to draw their character images. It took hours to scan them, many students based their drawings on online images anyway and some students didn’t participate as much as they felt they couldn’t draw well. The curriculum was altered to offer the choice of drawing game characters by hand or creating them using online .png images and a basic editing program such as Paint .NET.

Reference:
Bootstrap Curriculum, Unit 3 [11]
1. In the box below, write the Racket code (aka program) that defines a variable called `madness`. The variable will return "THIS IS SPARTA!". Please review the example:

```
> madness
THIS IS SPARTA!
```

2. Now, you must create a function that computes the perimeter of a square. You will specify the length of a side as input. Please see the examples below:

```
> (square-perimeter 4)
16
> (square-perimeter 100)
400
```

; square-perimeter : number ---&gt; number

(Created by Devin Cook)
PROGRAM DAY 7 – WEEK 2: DAY 2

Time Frame: 1 six-hour class day

Overview:
Students will learn how to design a function. Most of the day will be spent on Boolean as it is an important concept in games programming and computer science in general. Comparison and logical operators (AND, OR, NOT) will be covered.

Learning Objectives:
Students will be able to:
• Design a function
• Use Boolean logic

Assessments:
• In class observation

Materials:
• Bootstrap Workbook pp. 7, 10-12, 16-17
• Slides 7.1 – 7.18

INSTRUCTIONAL SEQUENCE

Instructional Procedures:

Review
• QUESTION: What did we learn yesterday?
• Bitmap String
• Put-image...

Speed Racer – Bootstrap Workbook p. 7
• Space students so no one sits together
• Students compute problems in lecture hall
• Faster examples of functions we used
• Sheet is blank, needs my problems:

<table>
<thead>
<tr>
<th>cost</th>
<th>price</th>
<th>number</th>
<th>;price + tax of 8.75</th>
</tr>
</thead>
<tbody>
<tr>
<td>days</td>
<td>years</td>
<td>number</td>
<td>;years * 365.25</td>
</tr>
<tr>
<td>alert</td>
<td>size</td>
<td>image</td>
<td>;red solid triangle</td>
</tr>
<tr>
<td>solid-circle</td>
<td>size color</td>
<td>image</td>
<td>;solid circle</td>
</tr>
<tr>
<td>solid-square</td>
<td>size color</td>
<td>image</td>
<td>;solid square</td>
</tr>
<tr>
<td>solid-star</td>
<td>size color</td>
<td>image</td>
<td>;solid star</td>
</tr>
</tbody>
</table>
• Discuss each and get student solutions

PAIR ACTIVITY: Design recipe
• Students design a function from scratch
• Partner with someone you don't know
• Rounds:
  o Rocket – Bootstrap Workbook p. 10
  o Red square – Bootstrap Workbook p. 11
  o Yard-Area – Bootstrap Workbook p. 12

SOLO: Bug Hunting
• Individual activity
• Find all bugs
• Can use computer to test
• Turn in the definitions window
• Must start with clean project – just #lang racket

Boolean Logic
• REVIEW: (+ 1 4), (/ 4 2), (- 0 9)
• QUESTION: What does (< 3 4) return?
• Boolean is EXTREMELY important in games

Boolean Functions
• Contributions of Boole
• Demo
  o I am holding a Pen
  o I am holding an Eraser
  o I am holding a Pen AND I am holding an Eraser (drop the pen)
  o I am holding a Pen AND I am holding an Eraser
  o I am holding a Pen OR I am holding an Eraser (drop the eraser)
  o I am holding a Pen OR I am holding an Eraser

Boolean: Powerpoint Presentation
• Slides 7.1 – 7.18

Worksheets again!
• Circle diagrams – Bootstrap Workbook p.16
• Butterfly – Bootstrap Workbook p.17
  o Reemphasize importance of Boolean in games
  o Boolean can also be used to check if two things hit
Closure:
- Daily Journal

Reflection:
The Bootstrap activities are turning out to be incomplete and somewhat confusing. The curriculum will move to Picturing Programs [12] based-activities beginning Day 9.

Reference:
Bootstrap Curriculum, Unit 4,6 [11]
Boolean Expressions

Are they fun? True!

Slide 7.1

Comparison Operator Overview

- Used to compare two data items
  - Equal, Unequal
  - Greater than, Less than
- Can be used with both numbers and strings
- Essential for writing complex programs

Slide 7.2
## Comparison Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equal To</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater Than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater Than or Equal</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less Than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less Than or Equal</td>
</tr>
</tbody>
</table>

### Examples

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>(= 1 4)</td>
<td>false</td>
</tr>
<tr>
<td>(&lt;= 24 31)</td>
<td>true</td>
</tr>
<tr>
<td>(&lt; 81 10)</td>
<td>false</td>
</tr>
<tr>
<td>(&gt;= 100 (+ 99 1))</td>
<td>true</td>
</tr>
</tbody>
</table>
Logical Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>Logical And</td>
</tr>
<tr>
<td>or</td>
<td>Logical Or</td>
</tr>
<tr>
<td>not</td>
<td>Unary Logical Negation</td>
</tr>
</tbody>
</table>

7.5

Logical Operators

- **OR Operator**
  - true if either operand is true
- **AND Operator**
  - true only if both operands are true
- **NOT Operator**
  - true if the operand is false
AND Example

$\text{true} \quad \text{true}$

$$\text{(and (\text{< 1 2}) (\text{< 10 40}))}$$

Result: true

Slide 7.7

AND Example 2

$\text{true} \quad \text{false}$

$$\text{(and (\text{< 1 2}) (\text{< 12 10}))}$$

Result: false

Slide 7.8
OR Example

true

\((\text{or} \ (< 1 \ 2) \ (< 10 \ 40))\)

Result: true

OR Example 2

true  false

\((\text{or} \ (< 1 \ 2) \ (< 12 \ 10))\)

Result: true
OR Example 3

\[(\text{or } (< 1 \ 2) \ (< 5 \ 10))\]

Result: false

Slide 7.11

NOT Example

\[(\text{not } (< 1 \ 2))\]

Result: false

Slide 7.12
NOT Example

\[(\text{not} \ (\ = 1\ 2))\]

Result: true

Slide 7.13

Examples

\[(\text{and} \ (< 1\ 3)\ (< 10\ 40))\]  true

\[(\text{and} \ (= 1\ 3)\ (< 10\ 40))\]  false

\[(\text{not} \ (= 1\ 2))\]  true

\[(\text{or} \ (> 1\ 3)\ (< 30\ 20))\]  false

Slide 7.14
If Statement

- This statement ...
  - executes a function if the expression is true
  - also contains a false branch
- Found in virtually every programming language

Basic Syntax

Either True or False

```
(if (Condition)
   (true part)
   (else part))
```
Example

(if (>= age 16)
   "Drive!"
   "Take the bus")

Example 2

(if (and (>= age 0)
         (<= age 120))
   "Valid"
   "Not human!")
PROGRAM DAY 8 – WEEK 2: DAY 3

Time Frame: 1 six-hour class day

Overview:
Today’s lesson reinforces math, logic and Boolean functions and introduces functions of functions.

Learning Objectives:
Students will be able to:
• Write functions
• Identify function components: input, output and function name
• Determine the domain and range of a function
• Compute what the output of a function will be with a given input
• Utilize functions in Racket to solve math problems

Assessments:
• In class observation

Materials:
• Student Bingo

INSTRUCTIONAL SEQUENCE

Instructional Procedures:
Review
• QUESTION: Comparison logic
• Boolean operators
• (if…) function

Functions in Functions
• Write out functions. Point out header and body.
  • Comparison logic
  • Boolean operators
  • (if ...) function

```
(define (f x)
  (- x 1))

(define (g y)
  (* y 2))
```
For "f" and "g":
- QUESTION: What is the function name?
- QUESTION: What is the input?
- Domain
- Range
- QUESTION: What's the output of: \( f(2) \)? \( f(5) \)? \( g(3) \), etc...
- QUESTION: What is the output?
  - \( f(g(2)) \)
  - \( g(f(4)) \)
  - \( f(g(f(10))) \)?

```scheme
(define (f x)
  (- x 1))

(define (g y)
  (* y 2))
```

**Even More Practice**

```scheme
(define (years-ago year)
  (- 2011 year))
```

- Using years-ago, create months-ago
- Using months-ago, create days-ago
- Using years-ago, create leap-years-ago

**More Math**
- sqrt
- expt

**Activity**
- Student Bingo

**Closure:**
- Daily Journal

**Reflections:**
Students enjoyed getting to know each other better during the Student Bingo activity.

**Reference:**
Bootstrap Curriculum, Unit 4,7 [11]
<table>
<thead>
<tr>
<th>Family</th>
<th>Arts</th>
<th>Recreation</th>
<th>Technology</th>
<th>International</th>
</tr>
</thead>
<tbody>
<tr>
<td>Born outside California</td>
<td>Has gone to the theatre this year</td>
<td>Likes to play basketball</td>
<td>Played <em>World of Warcraft</em></td>
<td>Has been to Europe</td>
</tr>
<tr>
<td>Has a pet dog</td>
<td>Has read all the Harry Potter books</td>
<td>Has gone skiing</td>
<td>Has had a computer virus</td>
<td>Has been to Asia</td>
</tr>
<tr>
<td>Has a pet cat</td>
<td>Can play a music instrument</td>
<td>Regularly rides a bike</td>
<td>Likes to buy and sell stuff on eBay</td>
<td>Born outside the United States</td>
</tr>
<tr>
<td>Is the youngest child in the family</td>
<td>Knows how to paint or sculpt</td>
<td>Has got a workout using Wii or Kinect</td>
<td>Has posted a video to YouTube</td>
<td>Is fluent in 2 or more languages</td>
</tr>
<tr>
<td>Has a younger sister / brother</td>
<td>Has sang in a choir</td>
<td>Has/wants to parachute from plane</td>
<td>Is the &quot;tech support&quot; for their friends</td>
<td>Enjoys Chinese Food</td>
</tr>
</tbody>
</table>
PROGRAM DAY 9 – WEEK 2: DAY 4

Time Frame: 1 six-hour class day

Overview: Students will learn more graphics functions and learn how to manipulate shapes to create images.

Learning Objectives:
Students will be able to:
• Use functions to manipulate images (including rotate, flip, copy, create patterns...)
• Crop Images

Assessments:
• In class observation

Materials:

INSTRUCTIONAL SEQUENCE

Instructional Procedures:
Review
• Functions in functions
• sqrt
• expt

Announcements
• Our lessons will now be based on Picturing Programs instead of Bootstrap
• You all are doing epically well
• Talk about all they have learned in 8 days!
  o The first day didn’t count
  o This is 4 weeks of college! (2 classes a week)
  o At 5 hours a day = 40 hours = 20 weeks!

Aligning Images
• Have students change (require Bootstrap) to (require picturing-programs)
• Delete (play ...)
• More graphics functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>flip-horizontal</td>
<td>Image → Image</td>
</tr>
<tr>
<td>flip-vertical</td>
<td>Image → Image</td>
</tr>
<tr>
<td>beside Image</td>
<td>Image → Image</td>
</tr>
<tr>
<td>above Image</td>
<td>Image → Image</td>
</tr>
<tr>
<td>overlay Image</td>
<td>Image → Image ;cheap version of place-image</td>
</tr>
</tbody>
</table>
• **PAIR ACTIVITY: Challenge-1-1**
  - What would you do if you wanted to see a picture, beside its left-to-right reflection?
  - Use one of your graphics
  - Player, goal, danger, etc...
  - Solution: Instead of pasting an image as one of the operands of the beside function, type in an expression involving flip-horizontal: 
    (beside IMG (flip-horizontal IMG ))

• **PAIR ACTIVITY: Challenge-1-2**
  - Rotating functions
  - Write an expression which displays a picture beside its top-to-bottom reflection
  
  ```
  rotate-cw : Image \rightarrow Image \\
  rotate-ccw : Image \rightarrow Image \\
  rotate-180 : Image \rightarrow Image \\
  rotate : Number Image \rightarrow Image 
  ```

• **PAIR ACTIVITY: Challenge-1-3**
  - Write an expression which displays a picture beside its 180-degree rotation

• **PAIR ACTIVITY: Challenge-1-4**
  - 2 x 2 square
  - Write an expression which displays four copies of a picture arranged in a 2 x 2 square
  - This might take a while
  - HINT: use definition window
  - HINT: break over several lines

• **PAIR ACTIVITY: Challenge-1-5**
  - Write an expression which displays four copies of a picture in a two-by-two square, each rotated differently: the top-right one should be rotated 90 degrees clockwise, the bottom-left one 90 degrees counter-clockwise, and the bottom-right one 180 degrees (i.e., same corner on each image is toward center of square)
Cutting Up Pictures

- Show new functions
  - Demo (crop-top (circle 100 "solid" "red") 50)
- QUESTION: Why this result?
- SOLO ACTIVITY: Eye
  - Create a variable called EYE
  - Draws a black eye with radius 20
- SOLO ACTIVITY: Mouth
  - Create a variable called MOUTH
  - Draws back half circle with radius 60
- SOLO ACTIVITY: FACE
  - Create a variable called MOUTH
  - Use eyes and mouth to get an odd face
- SOLO ACTIVITY: HAPPY
  - Overlay the face on yellow circle of radius 70

Text:

- Add "have a nice day" (or something else) to the picture
- Demo: NICE-DAY

Experimentation
- Demo: TEETH

Closure:
- Daily Journal

Reflection:
- Most students enjoyed today’s activities, especially creating the happy face.

Reference:
  - Picturing Programs, Chapter 1,3 [12]
PROGRAM DAY 10 – WEEK 2: DAY 5

Time Frame: 1 six-hour class day

Overview: Today will continue image manipulation and introduce colors, image functions and animation. The Friday enrichment activity here was a visit to Sac State’s anechoic chamber.

Learning Objectives:
Students will be able to:
• Explain the importance of and use comments in their code
• Overlay images
• Use RGB color function and transparency
• Save images they create
• Copy and pattern images using functions

Assessments:
• In class observation

Materials:

INSTRUCTIONAL SEQUENCE

Instructional Procedures:
Comments
• Importance of comments
• Line comments ;
• Block comments #| ... |#

Overlay/xy

```
overlay/xy : Image Number Number Image → Image
```

• Nearly identical to place-image
• QUESTION: How is this different?

Playing with Colors
• There are more than 16 million colors!
• ... but "red", "black" are quite limited
• RGB colors

```
make-color : Number Number Number → Image
```
• There is also a version with 4 colors.
• QUESTION: What does the 4th number do?
• ACTIVITY: Bubbles
• Create transparent circle called BUBBLE
  o place some translucent bubbles on your background

Saving your Work
• You can save any image you create
• It saves as a PNG
• So, you can save your happy face from last time

save-image : Image String \(\rightarrow\) Boolean

Function with images
• You can save any image you create
• PAIR ACTIVITY: (four-square...)
  o Define a function named four-square which, given an image, produces a two-by-two square of copies.
  o Try it on various images
• PAIR ACTIVITY: (checkered ...)
  o Define a function named checkered which, given two images, produces a two-by-two square with the first image in top-left and bottom-right positions, and the second image in top-right and bottom-left positions
  o Solution: The only new feature of this problem is that the function takes in two images rather than one with different parameter names:

\[
\text{(define (counterchange img1 img2)}
\text{ (above (beside img1 img2)
\text{ (beside img2 img1))}
\]

The Big Bang

(big-bang PLAYER (on-draw show-it))

• QUESTION: What happened?
• QUESTION: What the heck do you think "big-bang" means
• ACTIVITY:
  o Use place-image to put a picture on your background
  o Create a variable for that
  o Now, try it with the big bang
Enrichment Activity
• Visit to CSUS Physic Department’s anechoic chamber

Closure:
• Daily Journal

Reflection:
Students found the anechoic chamber interesting and a bit creepy.

Reference:
Picturing Programs, Chapter 3,4,6 [12]
PROGRAM DAY 11 – WEEK 3: DAY 1

Time Frame: 1 six-hour class day

Overview: Today will explore additional functions, colors and making images move.

Learning Objectives:
Students will be able to:
• Animate an image (using big-bang function)
• Make an image flip
• Write a tick handler
• Create a draw handler

Assessments:
• In class observation

Materials:

INSTRUCTIONAL SEQUENCE

Instructional Procedures:
A Few More Functions
• You used overlay/xy
• QUESTION: What the heck does it do?
• QUESTION: How is it different from place-image?

\[
\text{overlay/xy : Image Number Number Image } \rightarrow \text{ Image}
\]

More colors
• Make-color allows you to create any RGB color
• There is a version with FOUR numeric operators
• ACTIVITY: What does that fourth number do?

\[
\text{make-color : Number Number Number Number } \rightarrow \text{ Color}
\]

• ACTIVITY: Balloons
  o Make a function that takes a RGB and draws a circle with 128 alpha
  o DEMO: (balloon r g b)
  o Create a variable BIG-BALLOON – put on background
Function with images
- You can pass any image, number, string to a function
- So, let's make more general purpose functions
  - PAIR ACTIVITY: (four-square...)
  - PAIR ACTIVITY: (checkered ...)
- ACTIVITY: Frame-image
  - Frame-image Number Image → Image
  - Now create (beside-frame Number Image Image)
  - Now pad the eyes on your happy face

The Big Bang

```scheme
(big-bang PLAYER
  (on-draw show-it))
```

- QUESTION: What happened?
- QUESTION: What the heck do you think "big-bang" means
- ACTIVITY:
  - Use place-image to put a picture on your background
  - Create a variable for that
  - Now, try it with the big bang
- The first argument is the "initial" object

Tick Handlers
- QUESTION: How should the game update itself?
- (animate) called a function every 1/10 of second
- big-bang is far more sophisticated

```
[big-bang
  (overlay HAPPY (rectangle 200 200 "solid"
"white"))
  (on-draw show-it)
  (on-tick rotate-cw 0.25)]
```

- rotate-cw works since...
  - it takes in one argument of type Image
  - first argument is an Image
- on-draw supports a custom "resolution"
- ACTIVITY: Make a big-bang that makes your DANGER flip-horizontal every 1 second
- WARNING: Never leave out 'on-draw'. Your program will run, but not draw
Writing Tick Handlers

- Tick handler passes in the last image (since argument 1 is an image)
- You return a modified version of it
- That's why the image before was rotating

\[
\text{[define (my-tick pic) (underlay/xy pic 25 0 pic)]}
\]

\[
\text{[big-bang HAPPY}
\text{ (on-draw show-it 400 200)}
\text{ (on-tick my-tick 0.25)]}
\]

Draw Handlers

- show-it is just the default
- You can create your own with 'on-draw'
  - Works with any contract: image → Image
  - Racket passes in a blank image
  - You return the drawn one
- PAIR ACTIVITY: on-draw
  - Create your own draw function
  - Put the image on the background
  - Contract: my-draw : Image → Image
  - Image passed in is the HAPPY/tick image

\[
\text{[define (my-draw pic)}
\text{ (place-image pic}
\text{ 100}
\text{ 40}
\text{ (rectangle 600 500 "solid" "pink"))]}\]

\[
\text{[big-bang HAPPY}
\text{ (on-draw my-draw)}
\text{ (on-tick custom-tick 0.25)]}
\]

Event Handlers

- You can also call functions if an "event" happens
- QUESTION: What do I mean by event?
- You can handle key events too

\[
\text{[define (custom-key pic key)}
\text{ (underlay/xy HAPPY 25 0 pic)]}
\]

\[
\text{[big-bang PLAYER}
\text{ (on-draw show-it 500 100)}
\text{ (on-key custom-key)]}
\]
Closure:
• Daily Journal

Reference:
  Picturing Programs, Chapter 6,8 [12]
PROGRAM DAY 12 – WEEK 3: DAY 2

Time Frame: 1 six-hour class day

Overview: Big-bang is continued and strings are explored more extensively.

Learning Objectives:
Students will be able to:
• Use on-tick with Strings
• Create an on-draw handler
• Create a function that outputs an image

Assessments:
• In class observation

Materials:

INSTRUCTIONAL SEQUENCE

Instructional Procedures:
Models
• Big-Bang seems kinda weak
• ... that's because we are only using a bit of its power
• Big-Bang works on a "model" – number, image or a structure (we define later)
• This represents a "World"
• A World of images is very restrictive
• Big-Bang specifies...
  • model
  • events
  • handler – all input a model

; on-tick : model → model
; on-key : model key → model
; on-mouse : model number number mouse-event → model
; on-draw : model → image

• Only on-draw produces an image – that is what we see
• The rest modify the world
• Let's create a better World
Model of Number
• PAIR ACTIVITY: Make a function called Add-1
  o Input a number
  o Return the number + 1
• PAIR ACTIVITY: Make a function called Draw-Danger
  o Input a number
  o Return an image of your danger on a background
• PAIR ACTIVITY: Now, let's make it!!

```
(define (add-1 num) (+ num 1))
(define (draw-world num)
  (place-image danger num 100 BACKGROUND))
[big-bang 7
  (on-tick add-1 0.1)
  (on-draw draw-danger 500 400)]
```

• PAIR ACTIVITY: make a circle grow bigger each 1/10 second
• Congrats....
  o Your World consists of one number
  o Not very advanced, but we are getting better
• PAIR ACTIVITY: DANGER mouse!
  o Now create a handler for the mouse
  o Move the player whenever the mouse moves
  o Mouse events have 4 arguments

```
(define (draw-danger num)
  (place-image danger num 200 BACKGROUND))
(define (handle-mouse num x y mouse-event)
  (+ 1 num))
[big-bang PLAYER
  (on-draw draw-danger 640 480)
  (on-mouse handle-mouse)]
```

Bubble Gum!
• Our world increases its single num each time the mouse moves
• What we do with that number is up to us
• So, let's use it differently
• PAIR ACTIVITY: Create a function
  o Contract: bubble-gum : Number → Image
  o Draws a translucent pink circle with the inputted radius
• PAIR ACTIVITY: create an on-draw handler
  o Place-image in the center of your background
  o Make on-draw 640x480
  o Use your handle-mouse for on-mouse

Model of String
• A string can also be a model
• Let's create a World of String!
• Before, let’s review strings and cover a few more contracts

```
; string-length: String → Number
; string-append: String ... → Number
; text: String Number String → Image
```

• New ones

```
; substring : Str Num → Str ;start (0 index)
; substring : Str Num Num → Str ;start end
; string->number : Str → Num
; number->string : Num → Str
```

• PAIR: World of strings
  o Create a new handle-tick
  o It will take in a string, and return a string with "very" put in front of it
  o On-draw should draw it on a background (of your choice)
  o … best to place-image at 320x240
  o … and keep the font size somewhat reasonable - 12
  o Run it every 1

Closure:
• Daily Journal

Reference:
  Picturing Programs, Chapter 8,9 [12]
PROGRAM DAY 13 – WEEK 3: DAY 3

Time Frame: 1 six-hour class day

Overview: Program organization, use conventions and ASCII are introduced. Booleans are continued.

Learning Objectives:
Students will be able to:
• Reiterate the importance of use conventions and formatting
• Employ TeachScheme language conventions
• Explain why code reuse is significant
• Compare Strings using ASCII values

Assessments:
• In class observation

Materials:
• Slides 13.1 – 13.5

INSTRUCTIONAL SEQUENCE

Instructional Procedures:

Writing programs
• Programs can get large and complex
• And they can change midway
• You can get an assignment – and it changes on its due date!
• So, you need to write programs
  o that are easy to change
  o and are easy to read

Use Conventions
• Write variables in ALL-CAPS. This is standard
• Write functions in lower-caps.
• Put dashes between words

Use formatting
• Use multiple lines – Racket helps
• Change delimiters often (…) { …} […]

Break the program into smaller pieces
• Never write the same code twice
• If you did, you need a new function
This makes programs easier
  o debug – not redundant code
  o readable – function call is smaller than the code

Premature optimization is the root of all evil
  • Never optimize until your programs works
  • Then make a backup first
  • Never optimize if the program is hard to understand

```scheme
; Miles / mpg * price of gas
[define (gas-cost miles)
  (* (/ miles 24) 4.139)
]
[define (gas-cost miles)
  (* miles 0.1724583)
]
```

Back to Booleans
  • It’s time to go back to Boolean
  • Most of your program consists of True/False statements
  • Naming convention for Boolean values/functions
  • We need to become experts

Review of number comparison

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type 1</th>
<th>Type 2</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Number</td>
<td>Number</td>
<td>Boolean</td>
</tr>
<tr>
<td>&gt;</td>
<td>Number</td>
<td>Number</td>
<td>Boolean</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Number</td>
<td>Number</td>
<td>Boolean</td>
</tr>
<tr>
<td>&lt;</td>
<td>Number</td>
<td>Number</td>
<td>Boolean</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Number</td>
<td>Number</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

Oddly, there is no "not-equal" function

Review of number Boolean functions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type 1</th>
<th>Type 2</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>Boolean</td>
<td>Boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>or</td>
<td>Boolean</td>
<td>Boolean</td>
<td>Boolean</td>
</tr>
<tr>
<td>not</td>
<td>Boolean</td>
<td></td>
<td>Boolean</td>
</tr>
</tbody>
</table>

GROUP ACTIVITY: Let's make a "not-equals" function
  o Design what it should look like
  o Give C, Visual Basic, Prolog examples
- Take tally of notation

**Boolean String Functions**

<table>
<thead>
<tr>
<th>; string=?  : string string -&gt; Boolean</th>
</tr>
</thead>
</table>

- **PAIR ACTIVITY:** create (is-empty? string)
  - Return true if the string is empty
  - Otherwise false
  - You can use string=? or string-length
- Strings are compared using their ASCII value
- They are case-sensitive
- **PAIR ACTIVITY:** create (is-ipod string) → Boolean
  - true if the input is "ipod"
- **PAIR ACTIVITY:** modify function to return true for either "ipod", "ipad", or "iTouch"

**SLIDES: ASCII**

- lowercase letters are > uppercase ones

| ; string<?  : string string -> Boolean |
| ; string<=? : string string -> Boolean |
| ; string?>  : string string -> Boolean |
| ; string>=? : string string |

- **GROUP ACTIVITY:** string-not-equals
  - take tally of notations
- Racket provides case-insensitive functions

| ; string-ci=? : string string -> Boolean |
| ; string-ci<? : string string -> Boolean |
| ; string-ci<=? : string string -> Boolean |
| ; string-ci>?  : string string -> Boolean |
| ; string-ci>=? : string string -> Boolean |

- **PAIR ACTIVITY:** modify ipod function to be case-insensitive

**Closure:**
- Daily Journal

**Reference:**
- Picturing Programs, Chapter 8,1,13 [12]
Comparing Strings

- Racket contains functions for comparing strings
- ... but they don't quite behave the way you think they should
- Racket compares each character – one at a time
- It uses each character's ASCII value

Slide 13.1

ASCII Chart Review

Slide 13.2
Comparing Letters

<table>
<thead>
<tr>
<th>A</th>
<th>01000001</th>
<th>65</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>01000010</td>
<td>66</td>
</tr>
<tr>
<td>C</td>
<td>01000011</td>
<td>67</td>
</tr>
<tr>
<td>D</td>
<td>01000100</td>
<td>68</td>
</tr>
<tr>
<td>E</td>
<td>01000101</td>
<td>69</td>
</tr>
<tr>
<td>F</td>
<td>01000110</td>
<td>70</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a</th>
<th>01100001</th>
<th>97</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>01100010</td>
<td>98</td>
</tr>
<tr>
<td>c</td>
<td>01100011</td>
<td>99</td>
</tr>
<tr>
<td>d</td>
<td>01100100</td>
<td>100</td>
</tr>
<tr>
<td>e</td>
<td>01100101</td>
<td>101</td>
</tr>
<tr>
<td>f</td>
<td>01100110</td>
<td>102</td>
</tr>
</tbody>
</table>

"A" < "a"

Comparison Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>string=?</td>
<td>Equal To</td>
</tr>
<tr>
<td>string&gt;=?</td>
<td>Greater Than</td>
</tr>
<tr>
<td>string&gt;=?</td>
<td>Greater Than or Equal</td>
</tr>
<tr>
<td>string&lt;=?</td>
<td>Less Than</td>
</tr>
<tr>
<td>string&lt;=?</td>
<td>Less Than or Equal</td>
</tr>
</tbody>
</table>
## String Examples

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>(string=? &quot;a&quot; &quot;A&quot;)</td>
<td>False</td>
</tr>
<tr>
<td>(string&gt;? &quot;a&quot; &quot;A&quot;)</td>
<td>True</td>
</tr>
<tr>
<td>(string&lt;? &quot;abC&quot; &quot;abc&quot;)</td>
<td>True</td>
</tr>
<tr>
<td>(not (string=? &quot;dog&quot; &quot;cat&quot;))</td>
<td>True</td>
</tr>
</tbody>
</table>

Slides 13.5
PROGRAM DAY 14 – WEEK 3: DAY 4

Time Frame: 1 six-hour class day

Overview: Introduction to testing and program controls.

Learning Objectives:
Students will be able to:
• Create a function to stop a program
• Write a progress bar function

Assessments:
• In class observation

Materials:

INSTRUCTIONAL SEQUENCE

Instructional Procedures:
Testing Types
• Lots of errors happen when you pass a wrong type
• Fortunately, Racket has Boolean functions for that!

; number? : anything -> boolean
; image? : anything -> boolean
; string? : anything -> boolean
; boolean? : anything -> boolean
; integer? : anything -> Boolean

• PAIR ACTIVITY:
  o Use (integer?) to create a function called "(natural? ..)"
  o True for natural numbers 1, 2, 3, 4....
  o You will need an AND comparison operator too!

Booleans and Animations
• DISCUSSION: Let's make a program
• It will display Supercalifragilisticexpialidocious
• Each second it will delete one character off the front
  o QUESTION: What is the model?
  o QUESTION: What do we have to define?
    • chop-string
    • show-string
[define (chop-string model)
  (substring model 1)]

[define (show-string model)
  (place-image {text model 16 "yellow"
               100
               100
               BACKGROUND} ]

[big-bang "Supercalifragilisticexpialidocious"
  (on-draw show-string)
  (on-tick chop-string 1)]

• Problem with above
  o QUESTION: What happens when it gets to an empty string?
  o QUESTION: Let's see what happens...
  o QUESTION: How do we fix this?

• Repair
  o Big-bang can automatically stop
  o This uses a Boolean value
  o "stop-when" events happens AFTER on-draw
  o This is how NinjaCat stops

[define (empty-string? model)
  (string=? model "") ]

[big-bang "Supercalifragilisticexpialidocious"
  (on-draw show-string)
  (on-tick chop-string 1)
  (stop-when empty-string?)]

• PAIR ACTIVITY: Change the stop
  o Stop when you get to 3 characters
  o Write a new function

Using stop-when with Numbers
• PAIR ACTIVITY: bubble-gum!
  o Time to re-use the bubble-gum example
  o big-bang with add-1
  o Write a new function to stop when the bubble-gum gets as big as the background
• So, when it touches the edge, it stops
  o Your background is 640x 480

• PAIR ACTIVITY: Progress bar

; (overlay/align string string image image) → image

• Let's make another World of number
  o Write a function for a program-bar: contact: (progress-bar
    width height done-width color) → image
  o Write draw function : draw-progress-bar

**Closure:**
• Daily Journal

**Reflection:**

**Reference:**
  Picturing Programs, Chapter 8,13,14 [12]
PROGRAM DAY 15 – WEEK 3: DAY 5

Time Frame: 1 six-hour class day

Overview:
Today is an enrichment activity day. Students will take a practice ACT exam. Half of the students will take the exam in the morning and half in the afternoon. Students not taking the exam will participate in a Pong Tournament.

Learning Objectives:
Students will:
• Gain familiarity with the ACT test
• Understand individual strengths and weaknesses on the ACT and become motivated to study for the exam

Assessments:
• In class observation

Materials:
• Slides 15.1 – 15.22

INSTRUCTIONAL SEQUENCE

Instructional Procedures:

INTRODUCTION: Birth of Video Games – The History of Fun
• Slides 15.1 – 15.22

Enrichment Activity
• ACT Practice Test

Bonus Activity
• Pong Tournament

Closure:
• Daily Journal

Reflection:
Many students wrote about their plans to study for the ACT in their journals; this turned out to be a motivating activity. The Pong Tournament was a success. Students expressed in their journals that this was a great day.
The History of Fun

Slide 15.1

Magnavox Odyssey

- Created by Ralph H. Baer in 1972
- First video game console
  - chips made it affordable for the average consumer
  - wasn't a big success
- It is not a computer
  - analog with some digital components
  - does not contain a processor

Slide 15.2
Magnavox Odyssey

- **Features**
  - had "overlays" for your T.V. screen
  - had a light gun

- **Used "cards" for games**
  - these turned features on/off
  - but, these were variations of the same game
  - as a result, limited to original architecture

Show image of Odyssey system
Show image of Odyssey system

Slide 15.5

Pong

- Created in 1972 Nolan Bushnell
- Influenced by the Odyssey
- First used in a tavern in Sunnyvale, CA
  - next day, there was a line to play the game
  - the game had broken the night before
  - too many quarters jammed the machine!
- Beginning of the arcade game era

Slide 15.6
Show image of upright Pong machine
Pong Era Begins

- Atari created a home console!
  - first version was the C-100
  - could be hooked up to a standard television set
- A new form of entertainment
  - something the public had never seen before
  - this little device is creating a T.V. show that I can control!

Atari Teams Up with Sears

- Many stores would not sell Atari
  - didn't know what a "video game company" was
  - it was a weird product – for back then
- So, Atari teamed up with Sears
  - Atari created built them and Sears sold them under their name of Tele-Games
  - Atari also sold the same items under their name – with cosmetic changes
Pongs, Pongs and More Pongs!

- **Hundreds** of companies...
  - began to create pong consoles
  - it was a new, huge market
- General Instruments created a Pong "chip"
  - AY-3-8500 provided several games
  - could be put in a console, wired, with little cost
  - so, there were even **MORE** pongs!
Slide 15.13

Slide 15.14
Show image of Pong systems

Slide 15.15

AY-3-8500 Built in games

- Tennis (Pong)
- Hockey – sometimes called soccer
- Squash – sometimes called handball
- Practice (one player Handball)
- Target – gun game – bounces
- Skeet – gun game – flies across the screen

Slide 15.16
Atari Counter-attacks

- Decided to out-innovate the competition
  - created even more advanced models with different games
  - always used color
- Atari keep pushing the envelope
  - this would lead up to the end of the Pong Era and the rise of the modern video game era
  - ... and Atari would bring us there

Show image of Atari Super Pong
Image of Tele-Games Super Pong

Slide 15.19

Slide 15.20
PROGRAM DAY 16 – WEEK 4: DAY 1

**Time Frame:** 1 six-hour class day

**Overview:** Students will learn if/then and conditional programming structures and proper use of parentheses (curly vs. square).

**Learning Objectives:**
Students will be able to:
- Use the if function
- Employ the cond function

**Assessments:**
- In class observation

**Materials:**

INSTRUCTIONAL SEQUENCE

**Instructional Procedures:**

**Review stop-when**
- PAIR ACTIVITY:
  - Use stop-when on your progress bar
  - Now create stop-when to end when it is full

**Forcing a stop**
- Sometimes we want to stop after another event takes place
- Racket has a built in stopping function

```
; stop-with : model → model ; you give it a final
```

- PAIR ACTIVITY: Change Progress-Bar
  - End if a key is pressed
  - So, it will stop when it is full or the user hits a key

```
(define (stop-on-key model key)
 (stop-with model)
)
```
Review of the If Function

- Conditional expressions
- Selects one of two paths

```lisp
(if : Boolean Any Any → Any
(if (string=? name "Mr. Poo")
 "Awesome"
 "Not quite")
```

- PAIR ACTIVITY: Create a function that returns "Natural" or "Not Natural" depending if the number is a natural number

Good de-facto standard

- Only use parenthesis or curly if you are calling something
- Use square for the rest

```
cond Expression
```

- This will hurt….

```
(cond [ question answer1 ]
  [ question2 answer2 ]
  [ question3 answer3 ]
  ...
  [ else answer ]
)
```

- PAIR ACTIVITY:
  - Create a program that displays either your player, goal, etc...
  - User will pass in "goal", "target", etc...
  - DEMO: (choose-image String)

Else clause – what happens if we enter a wrong value?

- PAIR ACTIVITY: Number → Name
  - Convert a number 0 to 9 to its name
  - Use one conditional expression
  - DEMO: (number-name String)

- PAIR ACTIVITY: Name → Number
  - Now, convert a number's name to its value
  - Use one conditional expression
  - Make it case-insensitive!
  - DEMO: (number-value String)
• PAIR ACTIVITY: Natural numbers
  o Make a function that returns a message for a number – either "Not natural" or "Natural"
  o You can reuse your function from before
  o DEMO: (natural? Num)
• PAIR ACTIVITY: Colors
  o Make a function that returns colors of your own design
  o Use (make-color …)
  o DEMO: (my-color String)

Closure:
• Daily Journal

Reference:
  Picturing Programs, Chapter 14,15 [12]
PROGRAM DAY 17 – WEEK 4: DAY 2

Time Frame: 1 six-hour class day

Overview: Conditionals will be explored further.

Learning Objectives:
Students will be able to:
- Use conditional ordering logic
- Nest conditional statements
- Create an animation using conditional decision making

Assessments:
- Pop quiz
- In class observation

Materials:
- Quiz

INSTRUCTIONAL SEQUENCE

Instructional Procedures:

Review
- DEMO: (need-by-name String)

```
(cond [(string=? greeting "morning")   "Coffee"]
    [(string=? greeting "afternoon") "Nap"]
    [(string=? greeting "evening")   "Pizza"]
    [(string=? greeting "night")     "Sleep"])
```

- PAIR ACTIVITY: need-by-time
  - Pass in an hour 0 – 23
  - Return the same text (or similar) as (need-by-name)
  - Use the (and) function
  - Ranges:
    - night: 0 – 6
    - morning: 7 – 12
    - afternoon: 13 – 17
    - evening: 18 – 23
Pop Quiz
- Scatter students so no one sits together
- Anyone without a perfect score will be brought up to speed before lunch
- Talk about F's and college

Ordering cases in a conditional
- cond function stops when it gets to the first true item
- It goes in order from first to last
- You can do this to optimize code
- Yes – "premature optimization is the root of all evil"
- But, this is pretty easy

```lisp
(define (lunch money amount)
  (cond [(> amount 230) "iTouch"]
       [(> amount 180) "Nano 16GB"]
       [(> amount 150) "Nano 8GB"]
       [(> amount 50)  "Shuffle"]
       [else "Food"]
  )
)
```

- PAIR ACTIVITY: Redo need-by-time
  - Comment out the old version
  - New version needs to use waterfall logic

Nested Conditionals
- Conditionals can be put in conditionals
- This will occur with more complex logic
- Your game will use it
- Example: fruit/vegetable by color
- ACTIVITY: Classify
  - Remember (string?) (image?), etc...?
  - Write a function called "classify"
  - It will return a string depending on the type of data
  - Here's the trick – for numbers, return "number" or "real".
  - You need nested conditionals

```lisp
; classify : anything → string
```
Animations making decisions
- You can use conditions with your (on-tick) handler to use logic to update
- PAIR ACTIVITY: traffic light (string)
  - Make it change from red-yellow-green every few seconds
  - First: work on function (change-light)
    - green → yellow
    - yellow → red
    - red → green
  - Now write an (draw-light) function. Just draw a solid circle.
  - Now let’s make a World of String

Closure:
- Daily Journal

Reference:
  Picturing Programs, Chapter 15,17 [12]
1. Write a conditional expression that checks if a variable called num is positive. It will return either "Positive" or "Why be so negative". Don't worry about zero.

2. In the box below, write a function that inputs string (representing the name of a fruit) and outputs a string with its color. Use "cherry" → "red", "banana" → "yellow", "pickle" → "green"

```plaintext
; fruit-color : String --- String
```

(Created by Devin Cook)
PROGRAM DAY 18 – WEEK 4: DAY 3

Time Frame: 1 six-hour class day

Overview: Mouse and key handlers will be covered.

Learning Objectives:
Students will be able to:
• Employ mouse handlers to move player
• Use key handlers
• Create a paint program using mouse handlers

Assessments:
• In class observation

Materials:

INSTRUCTIONAL SEQUENCE

Instructional Procedures:
Animations making decisions
• PAIR ACTIVITY: Traffic light 2
  o Update your draw-light to use conditions
  o Make a classic 3-high traffic light
  o Unlit circles must be black
  o HINT: create three functions – draw-red, draw-green, draw-yellow
• PAIR ACTIVITY: now make the light also change with a key-press

Mouse Handlers
• Now that you understand conditionals, let’s use them
• The mouse handler was mostly skipped
• Contract below:
  ; old-model x-number y-number event-string → new-model

• The fourth argument contains information about what happened
• Racket events:
  o "button-down"
  o "button-up"
  o "move" – cursor changed, button up
  o "drag" – cursor changed, button down
  o "enter" – cursor just entered the window
  o "leave" – cursor just left the window
• We can't do much with x and y just yet, our World is just one number
• PAIR ACTIVITY: players!!!
  o We can use x and y, but temporarily
  o Let's create a World of Image
  o The big-bang will start with your BACKGROUND
  o You don't have to create an on-draw
  o Create a mouse handler and place your PLAYER every time the mouse button is clicked

• PAIR ACTIVITY: sketch-pad
  o Create another World of Image
  o However, start with a blank white rectangle – not your background
  o Draw a dot ONLY when the "drag" event takes place
  o You just created a paint program!

Key Handlers
• You can also use conditional expressions with pressed keys.

```
; on-key handler
; old-model key-string \rightarrow new-model
```

• The key argument contains a description of the key pressed.

• PAIR ACTIVITY: spying on the event!
  o Create a World of Number
  o When the user presses a key, print the key string on the screen
  o Uses (text ....)

• PAIR ACTIVITY: using the key value
  o Use a conditional in your mouse event to add-1 or subtract-1 from the model
  o Look for "left" and "right"
  o (on-draw) will use the model as a x value
  o Place your PLAYER on the background

Closure:
• Daily Journal

Reference:
Picturing Programs, Chapter 17,18 [12]

Reflection:
Student enjoyed learning how to animate their characters.
PROGRAM DAY 19 – WEEK 4: DAY 4

**Time Frame:** 1 six-hour class day

**Overview:** Today is enrichment activity day. In the morning random numbers are discussed and a guest lecturer from the Computer Science department will give a talk on Factorials and Recursion. The afternoon activity is a campus tour.

**Learning Objectives:**
Students will be able to:
- Understand the basic principles of recursion

**Assessments:**
- In class observation

**Materials:**

**INSTRUCTIONAL SEQUENCE**

**Instructional Procedures:**

**Random**
- Random number generator function

**Enrichment Activity: Lecture – Factorials and Recursion**

**Enrichment Activity: Campus Tour**

**Closure:**
- Daily Journal

**Reflection:**
This was a four day week due to 4th of July holiday. Students enjoyed the Recursion lecture. The campus tour was successful – the only complaint was that the day was too hot!
PROGRAM DAY 20 – WEEK 5: DAY 1

Time Frame: 1 six-hour class day

Overview: Programming structures are discussed. Students spend most of the day working on moving characters in different directions using on-key handler.

Learning Objectives:
Students will be able to:
• Use key handlers to move characters
• Articulate the definition and importance of structures
• Create structures (records) in Racket

Assessments:
• In class observation

Materials:

INSTRUCTIONAL SEQUENCE

Instructional Procedures:
Review Key Handlers
• You can also use conditional expressions with pressed keys.

```scheme
; on-key handler
; old-model key-string → new-model
```

• The key argument contains a description of the key pressed.
• PAIR ACTIVITY: using the key value
  o Use a conditional in your mouse event to add-1 or subtract-1 from the model
  o Look for "left" and "right"
  o (on-draw) will use the model as a x value
  o Place your PLAYER on the background

Structures
• Just a single number will not work
• Most games are based on (x,y) coordinates
• So, let’s go BEYOND single values in the structures

• DISCUSSION: Why combine data?
• DISCUSSION: Examples of records
• Racket allows you to create records (called structures)

```
(define-struct struct-name (field-name-1 ... field-name-n ))
```

• This causes a TON of things to happen
• Racket creates functions for you – based on the structure

```
; make - struct : n objects -> struct
; struct - field : struct -> object
; struct ? : object -> boolean
```

• DEMO: Create a student structure
• ACTIVITY: functions
  o Create (full-name) function
  o Create grade function (use existing code)
  o Have them create constants for other students – USE CAPS

• DEMO: Create Point structure
• DEMO: Create (point->string)
• ACTIVITY: Create move-up function
  o Show how to return duplicate point: (same-stupid-point ...)
  o Function returns the same point with one less Y

```
; move-up : point → point
```

• ACTIVITY: Create the move-left, move-right, move-down
• ACTIVITY: Create a World of Point
  o Start at any point (center screen is best)
  o Draw your player

• ACTIVITY: Control the player
  o Handle the on-key event to move the player
  o Keys are: "up", "down", "left", "right"

• We need to stop the player from moving off screen
• DEMO: move-up
  • Use cond
  • Look at what decreasing y WILL do
  • If it’s less than zero, return a point with y set to zero
  • Show code

• ACTIVITY: create the rest, use them

**Closure:**
• Daily Journal

**Reference:**
  Picturing Programs, Chapter 18, 21 [12]
PROGRAM DAY 21 – WEEK 5: DAY 2

Time Frame: 1 six-hour class day

Overview: Sprite structures are discussed.

Learning Objectives:
Students will be able to:
• Draw a sprite
• Place characters where they want them to appear on the screen

Assessments:
• In class observation

Materials:

INSTRUCTIONAL SEQUENCE

Instructional Procedures:

Review concept of sprites
• sprite = image + location
• Anything that moves around during gameplay
• DISCUSSION: name some sprites in games
• DISCUSSION: what about NinjaCat?

Sprite Structure
• Have students create a structure for a sprite

[define-struct sprite (image point)]

• ACTIVITY: create draw-sprite
  o Draw a single sprite on the background
  o Don’t worry about moving – yet

• ACTIVITY: Create a World of Sprite
  o Start x, y doesn’t matter
  o Draw your player
  o Now create the world using danger
  o … and goal
  o DISCUSSION: how much changed?
• ACTIVITY: Control the player
  o Handle the on-key event to move the player
  o Keys are: "up", "down", "left", "right"

Closure:
• Daily Journal
PROGRAM DAY 22 – WEEK 5: DAY 3

Time Frame: 1 six-hour class day

Overview: Different ways to place a character on a background were explored. Students learned how to create a game world.

Learning Objectives:
Students will be able to:
• Create a game world structure
• Add sprites to the world

Assessments:
• In class observation

Materials:

INSTRUCTIONAL SEQUENCE

Instructional Procedures:
It’s Time for a "World"
• Our games have been very limited so far
• They were based on single objects
  o image
  o number
  o string
  o point
  o sprite
• Games require all of these
• So, let's create a New World!

World Structure
• We are going to, together, create a basic World structure
• You will expand this to create your game
• So, don't worry if this is too simplistic
• Have students create a structure for World

[define-struct world (player danger goal)]

• ACTIVITY: create draw-world
  o Draw the player on the background
  o Don't worry about moving – yet
  o Don't worry about the other two
  o It takes in a "world" and return a "world"
ACTIVITY: Create a World
  • Start x, y doesn't matter
  • Draw your player

Adding other sprites
  • This is going to hurt
  • Racket is no good at this – and the solution is ugly
  • To put A and B on the screen....
    • We must put A on the image created by putting B on Background
    • The code is ugly
    • But, we will make this better
  • DEMO: draw-world
    • Let students copy solution
    • Too complex for them to figure out at this stage

```scheme
(define (draw-world w)
  (place-image (sprite-image (world-player w))
    (point-x (sprite-point (world-player w)))
    (point-y (sprite-point (world-player w))))

  (place-image (sprite-image (world-player w))
    (point-x (sprite-point (world-player w)))
    (point-y (sprite-point (world-player w)))
    BACKGROUND-IMAGE)
  )
)
```

place-sprite
  • This code is ugly
  • It will get worse as we add more objects
  • Your sprite structure contains
    • image
    • point:  x, y
  • We can create a utility function
  • This will take advantage of our sprite structure and simplify code
  • Have students copy solution (still too complex)

place-image : sprite image  →  image

```scheme
(define (place-sprite s image)
  (place-image (sprite-image s)
    (point-x (sprite-point s))
    (point-y (sprite-point s))
    image)
  )
)
```
• ACTIVITY: better draw-world
  o Comment-out your old draw-world
  o Rewrite it using place-sprite
• DISCUSSION: How well did the utility function improve things?
• ACTIVITY: add the danger and goal

Closure:
• Daily Journal
PROGRAM DAY 23 – WEEK 5: DAY 4

Time Frame: 1 six-hour class day

Overview: Students learn how to move their characters using key-handlers, wrap and scroll functions and speed variables.

Learning Objectives:
Students will be able to:
• Make sprites move using keys
• Use wrap and scroll to move characters at relevant speeds

Assessments:
• In class observation

Materials:

INSTRUCTIONAL SEQUENCE

Instructional Procedures:

Moving your player
• Your player will be moved using the keyboard
• Fortunately, we can reuse your code

• ACTIVITY: implement skeleton on-key
  o Call it key-world
  o It will make a world, return a world
  o Simply, make a world using the sprites from the old world
  o It won't do anything yet

• ACTIVITY: make it move
  o You can use your existing "key-sprite"
  o You just need to modify ONE line in your key-world

Moving your danger
• Your player responds to the keyboard
• The other sprites move on their own
• DISCUSSION: how do we do this?

• ACTIVITY: create a skeleton on-tick called tick-world
  o Just return a new world, with old values for now
  o We will expand this soon
• ACTIVITY: Create a function that scrolls a sprite
  o Tie it to a move-up, down, etc...
  o Which one is up to you
  o Call from tick-world on danger

  \[
  \text{scroll-sprite : sprite} \rightarrow \text{sprite}
  \]

• DISCUSSION: Why did they stop?

• ACTIVITY: Create wrap-left, right, etc....
  o Rather than locking at one side, move to opposite
  o Hey, we have a game

Getting Better Animation
• Everything moves at the same speed!
• That's not very cool
• Let's modify wrap-xxx and scroll-sprite to pass in a speed rather than using the global SPEED

  \[
  \text{scroll-sprite : sprite number} \rightarrow \text{sprite}
  \]

  \[
  \text{wrap-xxx : point number} \rightarrow \text{point}
  \]

• ACTIVITY: create it

Closure:
• Daily Journal
PROGRAM DAY 24 – WEEK 5: DAY 5

Time Frame: 1 six-hour class day

Overview: Today’s Enrichment Activity is meeting student mentors.

Learning Objectives:

Assessments:

Materials:

INSTRUCTIONAL SEQUENCE

Instructional Procedures:
Enrichment Activity: Meeting Mentors
PROGRAM DAY 25 - 29 – WEEK 6

Time Frame: 5 six-hour class days

Overview:  
The last week of the course is spent finishing the games.

Learning Objectives:  
Students will be able to:  
- Incorporate collision handling into their games  
- Terminate game when player dies  
- Add score to program  
- Use programming skills learned to finish complete games

Assessments:  
- In class observation  
- Test

Materials:  
- Test – Version 1  
- Test – Version 2

INSTRUCTIONAL SEQUENCE

Instructional Procedures:  
Game Completion

Closure:  
- Daily Journal

Reflection:  
Students were excited to complete their games and proud of the final results.
1. Vocabulary: Match definition to its word. There will be some words left over.

i) ___ used by computers to position items on the screen. Normally we use the names "x' and "y".
A. programming language
B. coordinate plane
C. syntax
D. semantics
E. function
F. input
G. bellybutton lint
H. output
I. range
J. processor
K. infix
L. firstfix
M. sunflower
N. data type
O. cheese
P. domain
Q. event
R. prefix
S. handler
T. character
U. pemdas
V. string item
W. sprite

ii) ___ this term refers to a function that reacts to an event – such as a clock tick or a key being pressed.

iii) ___ these expressions put the operator first. Remember your parenthesis!

iv) ___ numbers, strings, images, and Booleans are all examples of this

v) ___ term for the value returned from a function.

vi) ___ these expressions put the operator between the two values. This is the format you use in algebra.

vii) ___ each letter, symbol, space, etc… that makes up a string

vii) ___ term for the values that you pass into a function.

ix) ___ graphical object on the screen that moves
x) _____ you write these to perform tasks such as draw your background, compute values, etc…
      Basically, anything you want the computer to do.

2. What is the name of the programming language you were using?
   a. Rack
   b. Racket
   c. Ratchet
   d. Rat
   e. Pemdas

3. In the box below, write the define for a variable called MESA. The variable will return "ROCKS!". Please review the example:

   > MESA
   ROCKS!

4. Which of the following is valid syntax for adding 1 and 2?
   a. [+ 1 2]
   b. (+ 1 2)
   c. {+ 1 2}
   d. all of the above
   e. a. and b.
5. Create a function that computes half the value of any number. Why? Why not!
   Please see the examples below:

   > (half 10)
   5
   > (half 42)
   21

   ; half : number ---\rightarrow\ number

6. Write an expression that checks if a variable called num is 0. If num is zero, it
   will be true. Otherwise, false. You don't have to write a whole function, Just show
   you know to compare values.
And now, for no apparent reason, here is a picture of a bunny with a pancake on its head.

7. When you create a structure, the system creates some functions for you. In the following code, what is the name of the function that gets the student's name from a variable called "a".

```
[define-struct student (name age grade-level)]
```

a. `(student (name a))`
b. `(name a)`
c. `(student-name a)`
d. none of the above
8. What is the result of the following Boolean expression?

\[
\text{(or (} > 1 \text{ 3) (< 30 20) )}
\]

a. True  
b. False  
c. None of the above  
d. All of the above  
   (which includes "None of the above"… wait, that makes no sense!)

9. Bubble gum! Yum! Write a function that inputs a radius and draws a pink, solid circle with that radius. I put the contract for circle below. You don't have to make it translucent, but that would be awesome!

; circle : number string string \(\rightarrow\) image

10. What is the result of the following mathematical expression?

\[
(/ (+ (* 4 16) (- 23 3)) 2)
\]

Result:
11. What is the result of the following Boolean expression?

\[
\text{(and } (< 1 \ 3) \ (< 10 \ 40))
\]

a. True  
b. False  
c. None of the above  
d. I have no clue!

12. Let's make an image! We have two variables designed: BOX draws a single square and DOT draws a circle. Use your knowledge of (beside) and (above) to create the image below.

\[
\text{; above : image } \rightarrow \text{ image}
\]
\[
\text{; beside : image } \rightarrow \text{ image}
\]
13. Just for fun: Hanna’s father has five daughters. The first four are named: Lala, Lele, Lili, Lolo. What is the name of the fifth daughter?

a. Leela
b. Lulu
c. Leia
d. Hanna
e. None of the above
Test Your Knowledge

Name:

1. What is the name of the programming language you were using?
   a. Rack
   b. Racket
   c. Ratchet
   d. Rat
   e. Pemdas

2. In the box below, write the define for a variable called MESA. The variable will return "ROCKS!". Please review the example:

   > MESA
   ROCKS!

3. Which of the following is valid syntax for adding 1 and 2?
   a. [+ 1 2]
   b. (+ 1 2)
   c. {+ 1 2}
   d. all of the above
e.  a. and b.
4.  Vocabulary: Match definition to its word. There will be some words left over.

i) ___ used by computers to position items on the screen. Normally we use the names "x' and "y".

ii) ___ this term refers to a function that reacts to an event – such as a clock tick or a key being pressed.

iii) ___ these expressions put the operator first. Remember your parenthesis!

iv) ___ numbers, strings, images, and Booleans are all examples of this

v) ___ term for the value returned from a function.

vi) ___ these expressions put the operator between the two values. This is the format you use in algebra.

vii) ___ each letter, symbol, space, etc… that makes up a string

vii) ___ term for the values that you pass into a function.

i) ___ graphical object on the screen that moves

x) ___ you write these to performs tasks such as draw your background, compute values, etc… Basically, anything you want the computer to do.
5. Write an expression that checks if a variable called `num` is 0. If `num` is zero, it will be true. Otherwise, false. You don't have to write a whole function, just show you know to compare values.

6. Create a function that computes half the value of any number. Why? Why not!
Please see the examples below:

```lisp
> (half 10)
5
> (half 42)
21
```

`; half : number --> number`
7. When you create a structure, the system creates some functions for you. In the following code, what is the name of the function that gets the student's name from a variable called "a".

[define-struct student (name age grade-level)]

a. (student (name a))
b. (name a)
c. (student-name a)
d. none of the above
8. Let's make an image! We have two variables designed: BOX draws a single square and DOT draws a circle. Use your knowledge of (beside) and (above) to create the image below.

![Image with a circle and a square beside each other]

; above : image \(\rightarrow\) image
; beside : image \(\rightarrow\) image

9. What is the result of the following Boolean expression?

\[
\text{and} \ (\text{<} \ 1 \ 3) \ (\text{<} \ 10 \ 40))
\]

a. True
b. False
c. None of the above
d. I have no clue!
10. Bubble gum! Yum! Write a function that inputs a radius and draws a pink, solid circle with that radius. I put the contract for circle below. You don't have to make it translucent, but that would be awesome!

```plaintext
; circle : number string string \rightarrow image
```

11. What is the result of the following mathematical expression?

```
(/ (+ (* 4 16) (- 23 3)) 2)
```

Result:

12. What is the result of the following Boolean expression?

```
(or (> 1 3) (< 30 20))
```

a. True
b. False
c. None of the above
d. All of the above

(which includes "None of the above"… wait, that makes no sense!)
13. Just for fun: Hanna’s father has five daughters. The first four are named: Lala, Lele, Lili, Lolo. What is the name of the fifth daughter?

a. Leela
b. Lulu
c. Leia
d. Hanna
e. None of the above
APPENDIX B

Data from Student Journal Analysis
Skills Perception Data

The following tables contain the number of ratings by student each class day for skills perception. This data was derived from the analysis of student daily journals.

### Skills Perception Ratings - Week 1 (40 students)

<table>
<thead>
<tr>
<th>Day:</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rating</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Great Improvement</td>
<td>21</td>
<td>23</td>
<td>29</td>
<td>15</td>
</tr>
<tr>
<td>Medium Improvement</td>
<td>8</td>
<td>8</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Little Improvement</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Skills Perception Ratings - Week 2 (40 students)

<table>
<thead>
<tr>
<th>Day:</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rating</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Great Improvement</td>
<td>18</td>
<td>25</td>
<td>24</td>
<td>26</td>
<td>22</td>
</tr>
<tr>
<td>Medium Improvement</td>
<td>5</td>
<td>8</td>
<td>6</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>Little Improvement</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

### Skills Perception Ratings - Week 3 (40 students)

<table>
<thead>
<tr>
<th>Day:</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rating</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Great Improvement</td>
<td>29</td>
<td>25</td>
<td>15</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td>Medium Improvement</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Little Improvement</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### Skills Perception Ratings - Week 4 (40 students)

<table>
<thead>
<tr>
<th>Rating</th>
<th>Day: 16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Great Improvement</td>
<td>15</td>
<td>12</td>
<td>22</td>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>Medium Improvement</td>
<td>12</td>
<td>9</td>
<td>4</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>Little Improvement</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

### Skills Perception Ratings - Week 5 (40 students)

<table>
<thead>
<tr>
<th>Rating</th>
<th>Day: 21</th>
<th>22</th>
<th>23</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Great Improvement</td>
<td>20</td>
<td>14</td>
<td>22</td>
<td>17</td>
</tr>
<tr>
<td>Medium Improvement</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Little Improvement</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

### Skills Perception Ratings - Week 6 (40 students)

<table>
<thead>
<tr>
<th>Rating</th>
<th>Day: 26</th>
<th>27</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td>Great Improvement</td>
<td>26</td>
<td>23</td>
<td>18</td>
</tr>
<tr>
<td>Medium Improvement</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Little Improvement</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Enjoyment Data
The following tables contain the number of ratings by student each class day for enjoyment. This data was derived from the analysis of student daily journals.

Enjoyment Ratings - Week 1 (40 students)

<table>
<thead>
<tr>
<th>Day:</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rating</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very Fun</td>
<td>12</td>
<td>15</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Mostly Fun</td>
<td>22</td>
<td>18</td>
<td>16</td>
<td>15</td>
</tr>
<tr>
<td>OK</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Mostly Boring</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Very Boring</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Enjoyment Ratings - Week 2 (40 students)

<table>
<thead>
<tr>
<th>Day:</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rating</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very Fun</td>
<td>8</td>
<td>7</td>
<td>13</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>Mostly Fun</td>
<td>20</td>
<td>29</td>
<td>18</td>
<td>9</td>
<td>14</td>
</tr>
<tr>
<td>OK</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Mostly Boring</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Very Boring</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Enjoyment Ratings - Week 3 (40 students)

<table>
<thead>
<tr>
<th>Day:</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rating</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very Fun</td>
<td>6</td>
<td>11</td>
<td>8</td>
<td>7</td>
<td>25</td>
</tr>
<tr>
<td>Mostly Fun</td>
<td>28</td>
<td>19</td>
<td>19</td>
<td>24</td>
<td>7</td>
</tr>
<tr>
<td>OK</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mostly Boring</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Very Boring</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### Enjoyment Ratings - Week 4 (40 students)

<table>
<thead>
<tr>
<th>Day:</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rating:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very Fun</td>
<td>4</td>
<td>12</td>
<td>11</td>
<td>17</td>
<td>3</td>
</tr>
<tr>
<td>Mostly Fun</td>
<td>22</td>
<td>16</td>
<td>22</td>
<td>14</td>
<td>23</td>
</tr>
<tr>
<td>OK</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Mostly Boring</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Very Boring</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Enjoyment Ratings - Week 5 (40 students)

<table>
<thead>
<tr>
<th>Day:</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rating:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very Fun</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Mostly Fun</td>
<td>22</td>
<td>19</td>
<td>24</td>
<td>20</td>
</tr>
<tr>
<td>OK</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Mostly Boring</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Very Boring</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Enjoyment Ratings - Week 6 (40 students)

<table>
<thead>
<tr>
<th>Day:</th>
<th>26</th>
<th>27</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rating:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very Fun</td>
<td>10</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>Mostly Fun</td>
<td>17</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>OK</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Mostly Boring</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Very Boring</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
REFERENCES


