Flashcard App

Jacob Cody Ho
Mary Young
Charlotte Ruelens
Franklin Sneider
Kramer Allen

Content

- Introduction
- Initial Proposals
- Structure of Apps
- Features of our App
- Demo
- Challenges
  - Data storage
  - Camera
  - Learning algorithm
- Cuts
- Conclusion

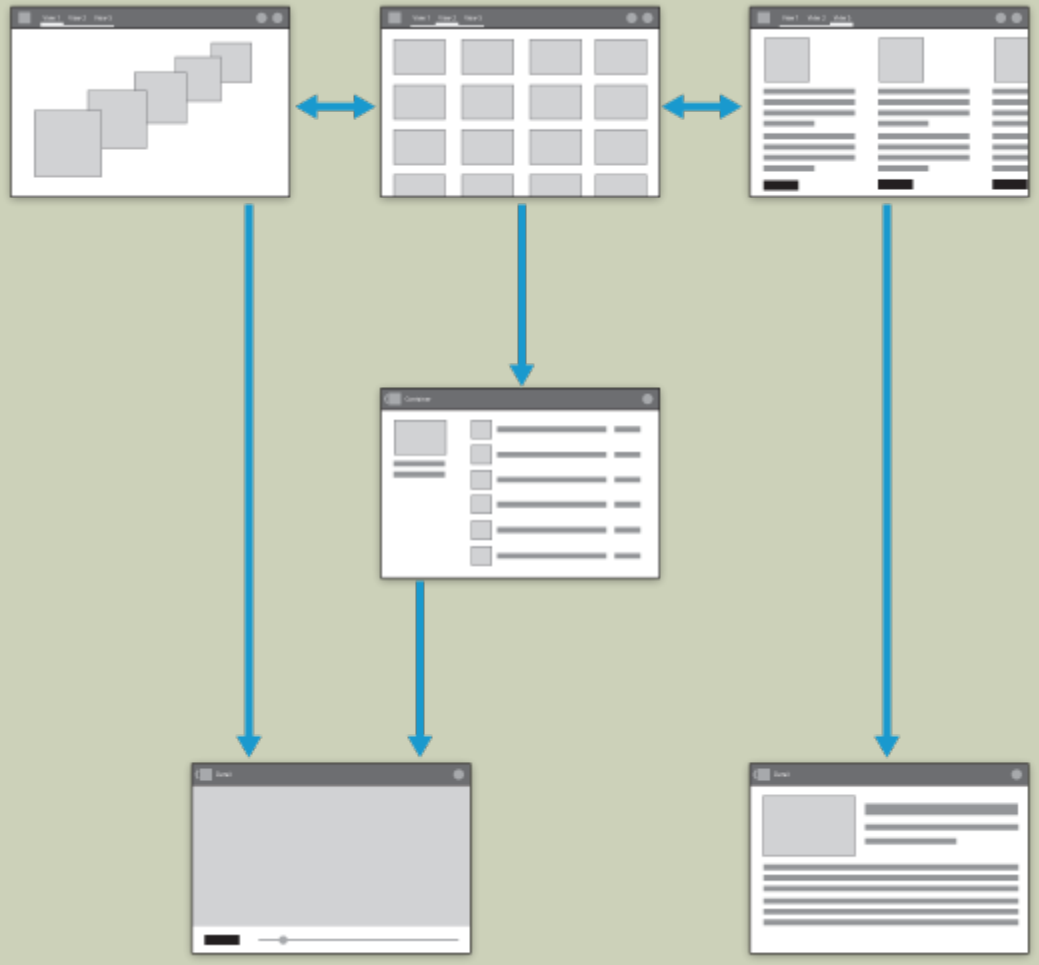
Introduction

- Why an Android App?
  - Important skill
  - Not taught in required courses
  - Popular mobile OS along with iOS
  - Fun!
- Why Flashcards?
  - Reasonable difficulty for novices
  - Practical for Vanderbilt students
  - Requires multiple activities/screens

Initial Proposal

- Android flashcard app with:
  - Multiple decks, cards with front and back
  - Organized data storage
  - Smart testing algorithm
  - Camera/touchscreen integration
- No existing app we found on Android store contains all features

Structure of Android App – User Perspective

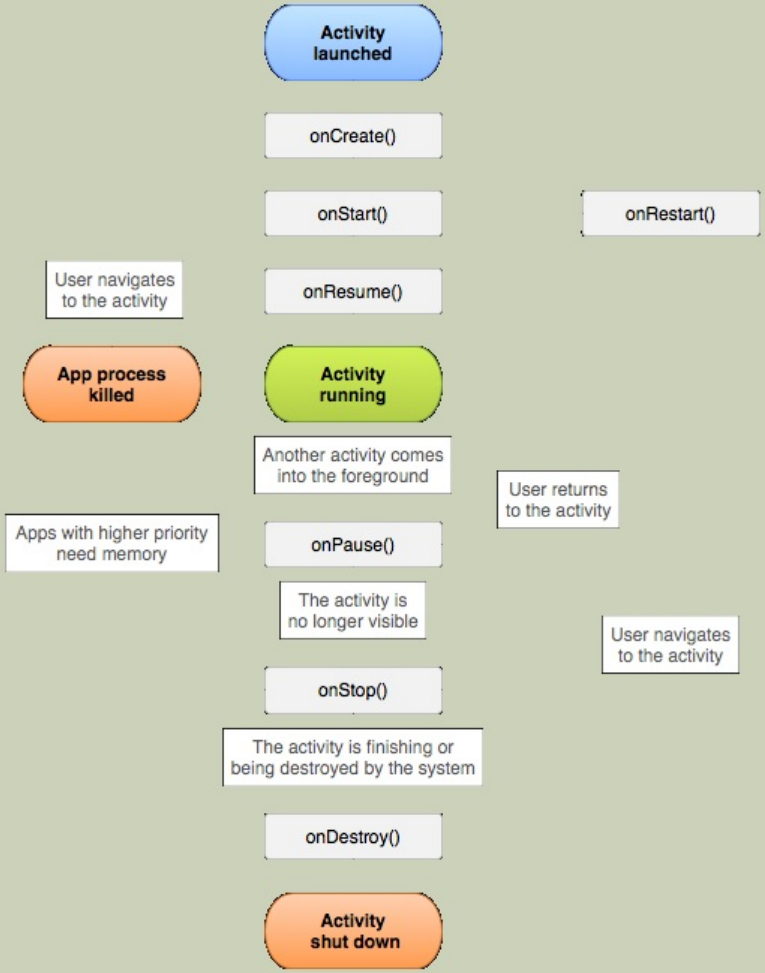


Top level views – expose the data

Category views – “zooms” in on a particular part of the data

Detail/Edit Views– allows you to edit or view

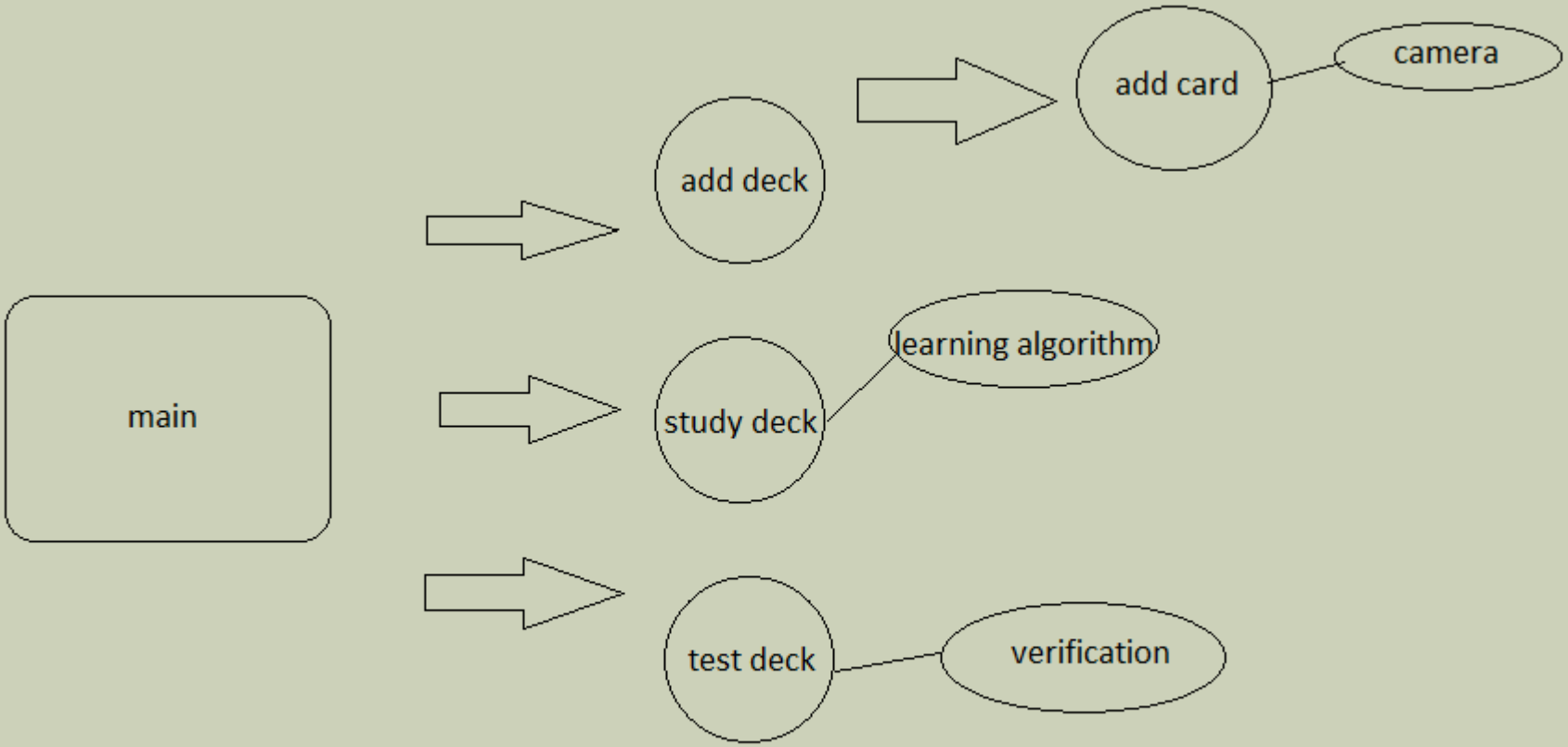
Structure of App – Developer Perspective



Final App Overview

- User can create/store multiple decks
- Camera integration
- Cards have a front and back
- Organized data storage
- Efficient testing algorithm
- Study mode
- Easy user experience

Structure of our App



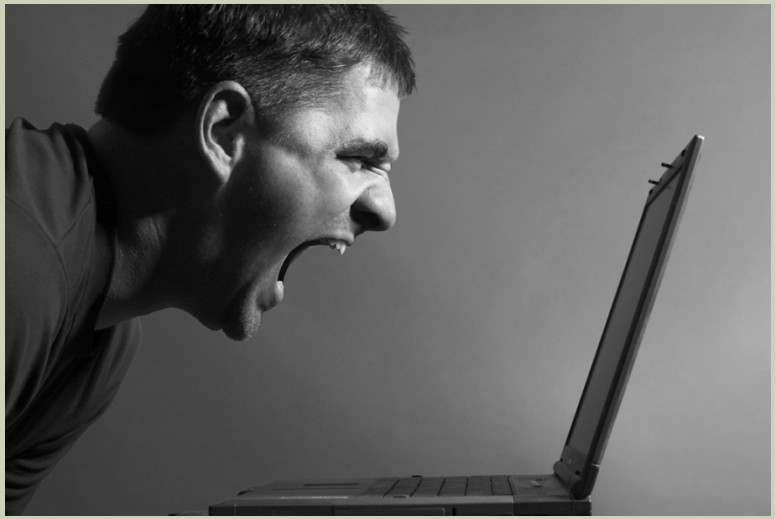
Structure of App

- Screenshots of App

DEMO

Biggest Implementation Challenges

- Android Novices
- Trouble with AVD
- Data Structure
- Camera Integration
- Learning Algorithm



Data Structure - Card

- Very simple data structure to represent a single card
- Chose not to store the Bitmap
- Instead, we keep track of if the back is an image and if it is the back is set to the file location
- Greatly simplified “study” mode programming
  - Data structure used to make Queues of cards

Card

Data Members

String front

String back

boolean hasPicture

Functions

Constructors

Getters

Setters

Deck Storage

- Main app has a deck list file, which contains all the names of the decks
- Each deck has a text file with the list of cards
- Folder with deck name containing image resources for the back of the cards
- Stored in external storage
  - SD card
- Each activity will read the text file and upload it into a list of Cards or strings as needed

Camera Integration

- Required learning existing API
- Launch a new activity with the camera
- Save into external storage
- Integrate into data structure

Camera Integration Code

```
protected void startCameraActivity() {
    // Create new file with name mentioned in the path variable
    File file = new File(path + Globals.CurrentDeckName + "/"
                + TextInFront.getText().toString() + "back" + ".jpg");


    if (!file.exists()) {
        img = TextInFront.getText().toString() + "back.jpg";
        System.out.println("image title: " + img);
        // Creates a Uri from a file
        Uri outputFileUri = Uri.fromFile(file);
        // Standard Intent action that can be sent to have the
        // camera application capture an image and return it.
        // You will be redirected to camera at this line
        Intent intent = new Intent(
                android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
        // Add the captured image in the path
        intent.putExtra(MediaStore.EXTRA_OUTPUT, outputFileUri);
        // Start result method - Method handles the output
        // of the camera activity
        startActivityForResult(intent, 0);
```

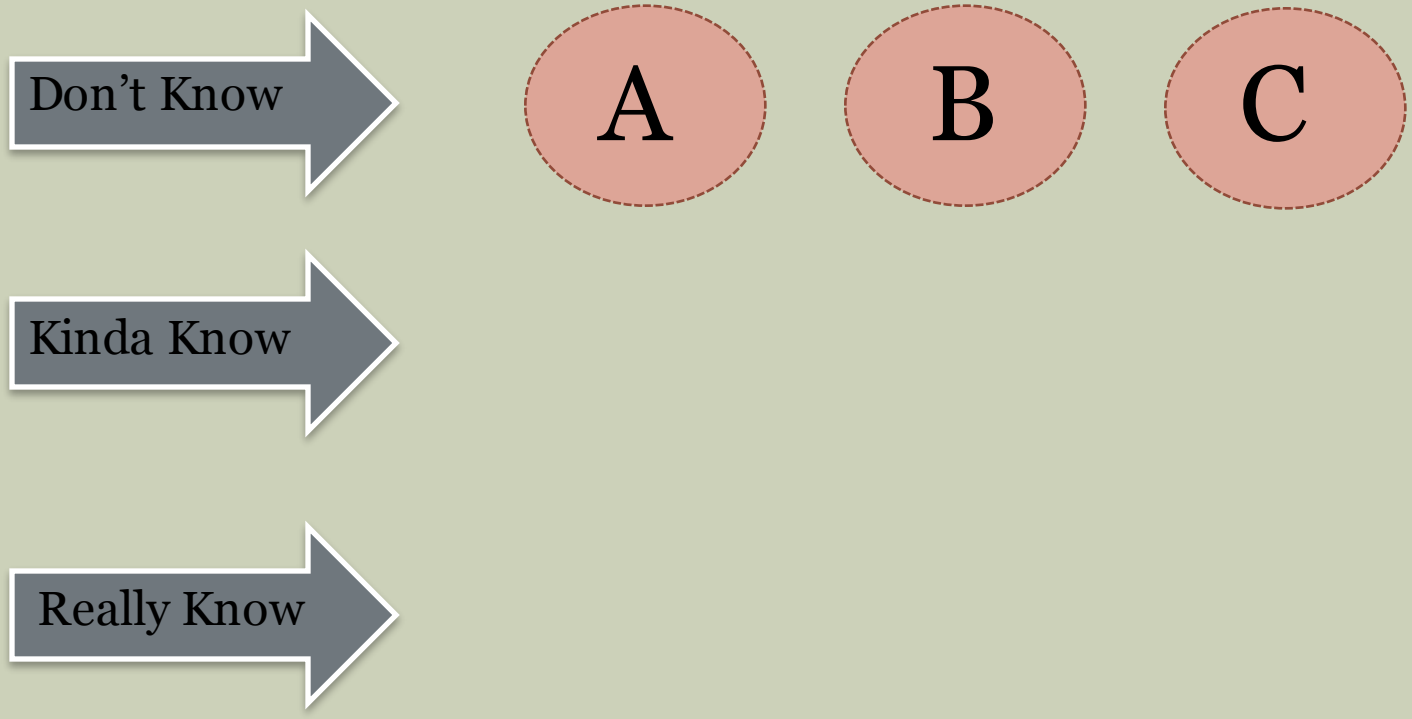
Learning Algorithm

- Criteria for algorithm
  - Somewhat random order of appearance
  - Rotate through all cards
  - Frequency of appearance based on knowledge
- Basic algorithm
  - Create three queues: one for don't know, one for kinda know, and one for really know
  - At the start, set all cards to don't know
  - Until the user quits, randomly select a queue such that don't know appears ½ of the time, kinda know appears 1/3 of the time, and really know appears 1/6 of the time.
  - Dequeue the first card in that queue and ask the user how well they know that card
  - Enqueue the card onto the appropriate queue
  - Continue

Learning Algorithm Code

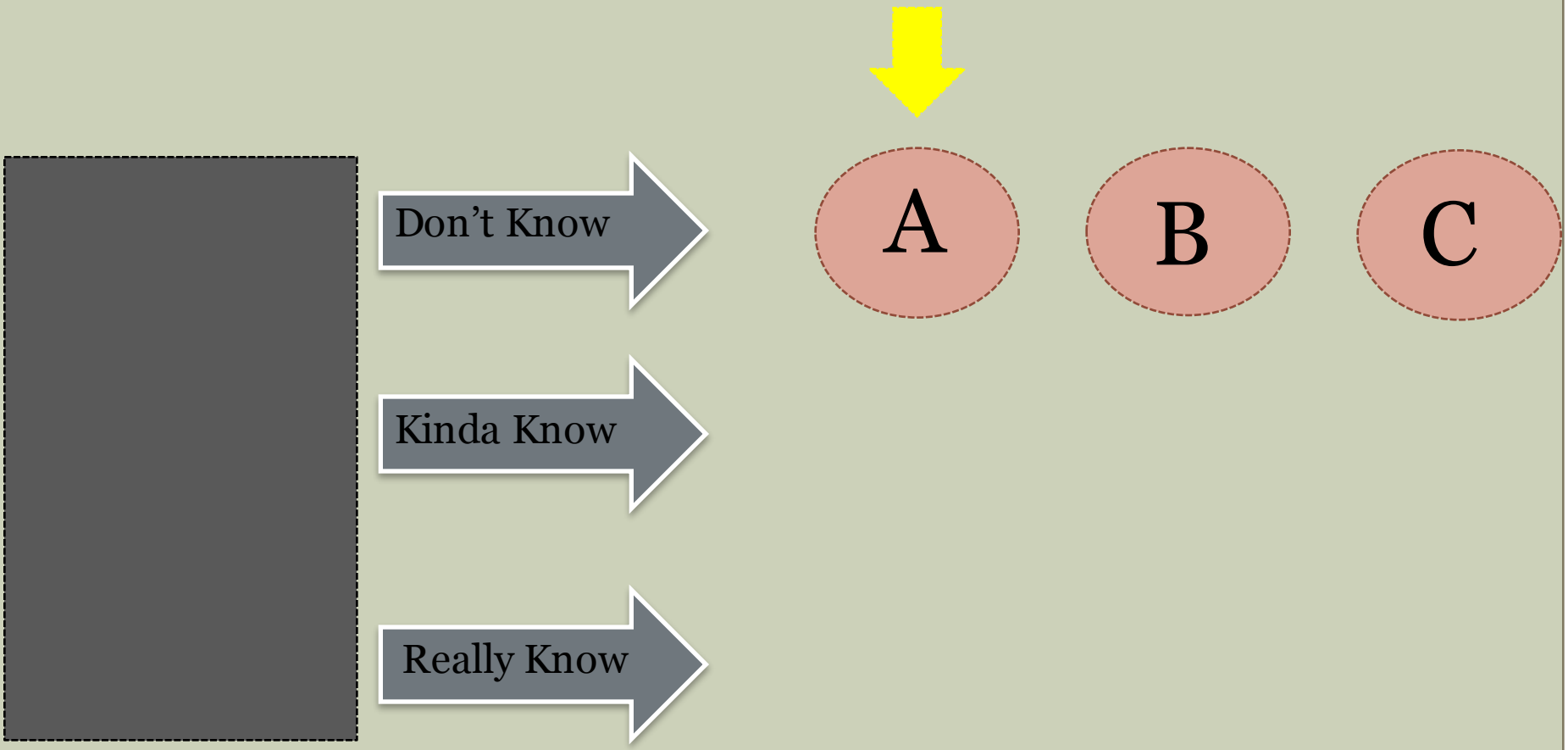
```
int R = generator.nextInt(6) + 1; //randomly generate a number for the queue

System.out.println("R = " + R);
// check if queues are empty
boolean found = false;
//While an appropriate queue hasn't been found, keep trying to get a number
while (!found) {
    //1-3 maps to don't know queue
    if (R <= 3) {
        if (!dontKnow.isEmpty()) {
            curCard = dontKnow.remove();
            x = 1;
            found = true;
        } else {
            R = generator.nextInt(6) + 1;
        }
    //4-5 maps to the kinda know queue
    } else if (R <= 5) {
        if (!kindaKnow.isEmpty()) {
            curCard = kindaKnow.remove();
            found = true;
            x = 2;
        } else {
            R = generator.nextInt(6) + 1;
        }
    } else {
    //6 maps to the really know queue
        if (!reallyKnow.isEmpty()) {
            curCard = reallyKnow.remove();
            found = true;
            x = 3;
        } else {
            R = generator.nextInt(6) + 1;
        }
    }
```

Learning Algorithm - Example



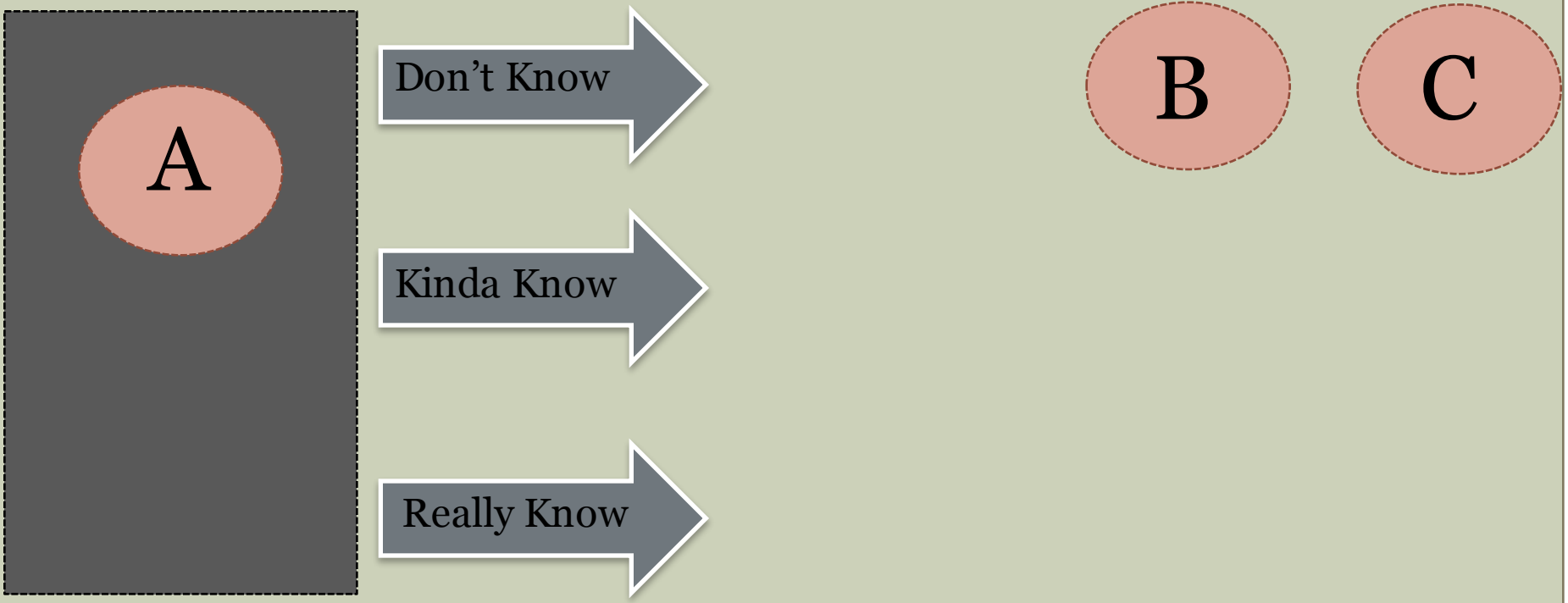
Learning Algorithm - Example

Action: Since there is only one queue, choose front of “Don’t Know” queue



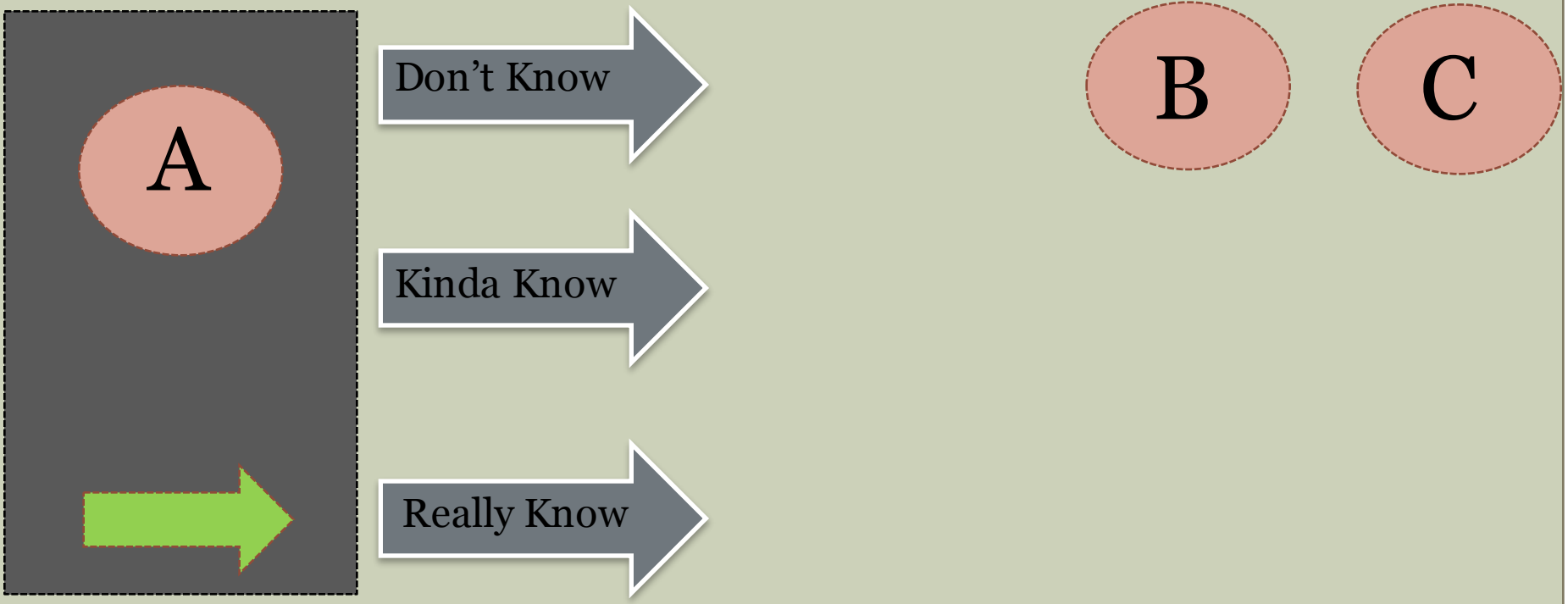
Learning Algorithm - Example

Action: A is sent to the screen, waiting for a user response



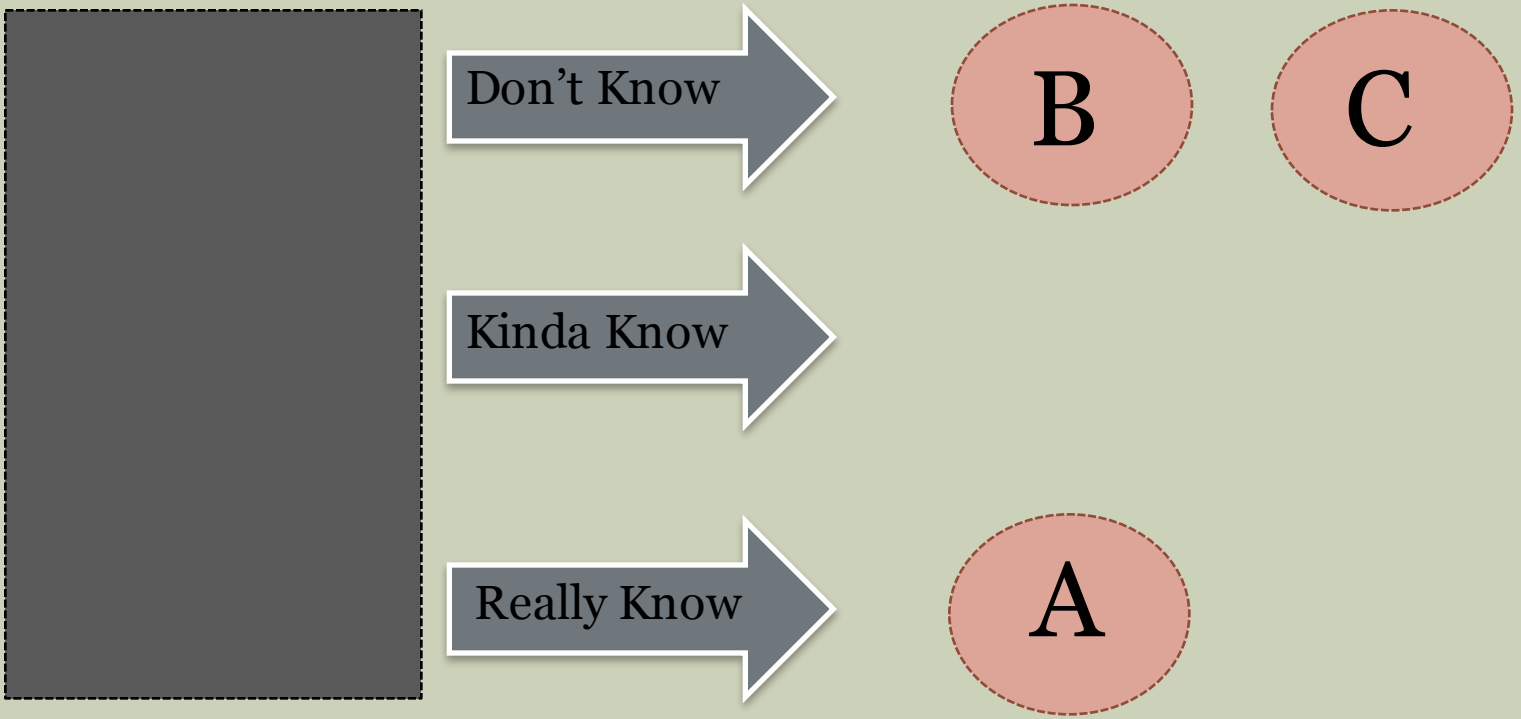
Learning Algorithm - Example

Action: User selects they really know the card now



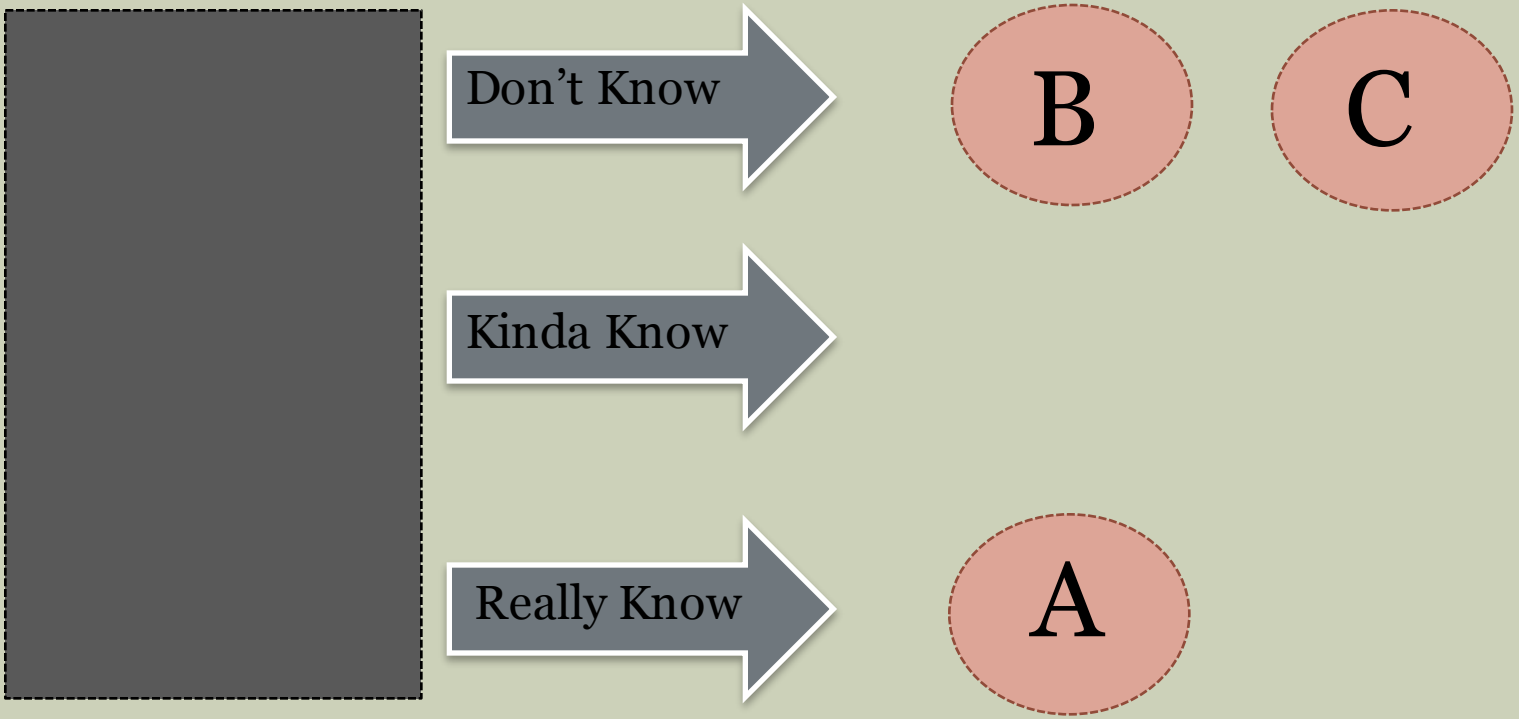
Learning Algorithm - Example

Action: A is moved to the Really Know Queue



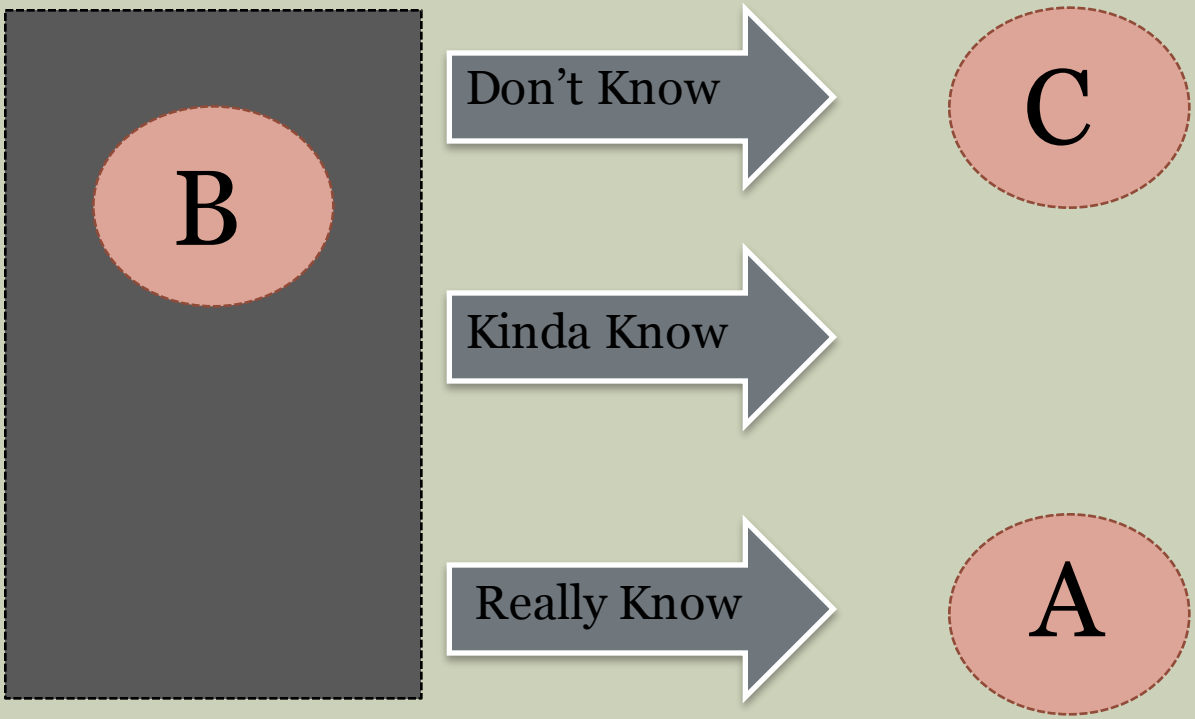
Learning Algorithm - Example

Action: The random generator generates a 3



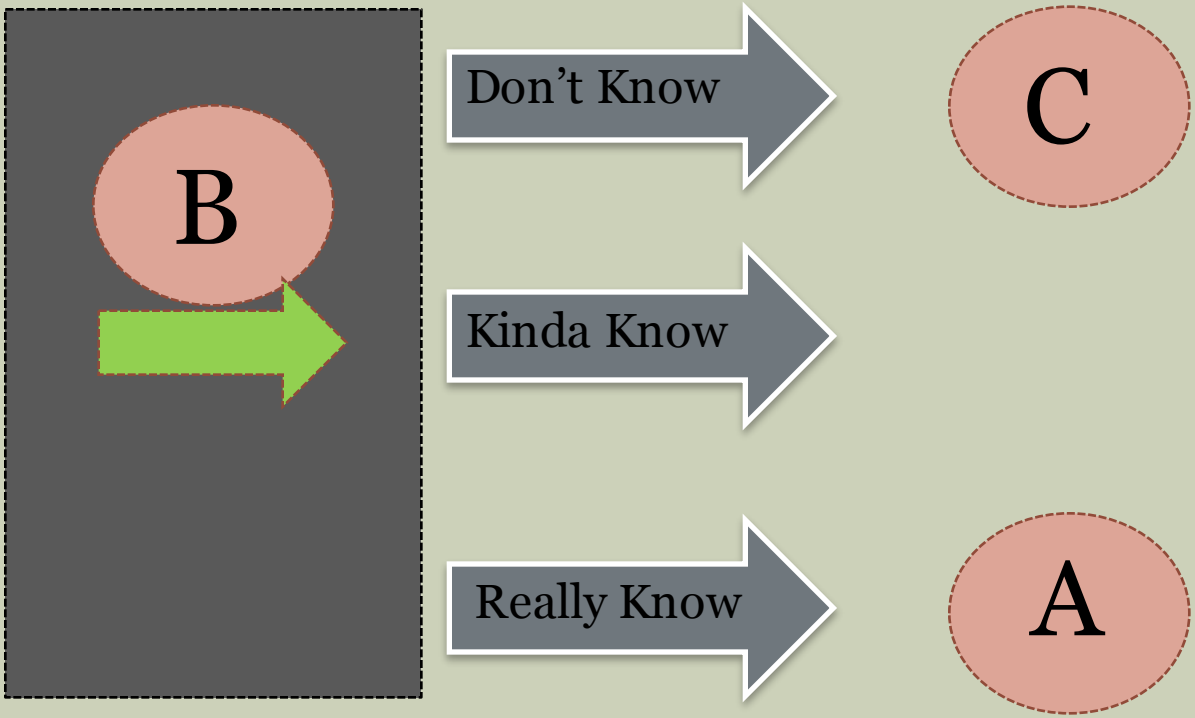
Learning Algorithm - Example

Action: B is sent to the screen, wait for user response



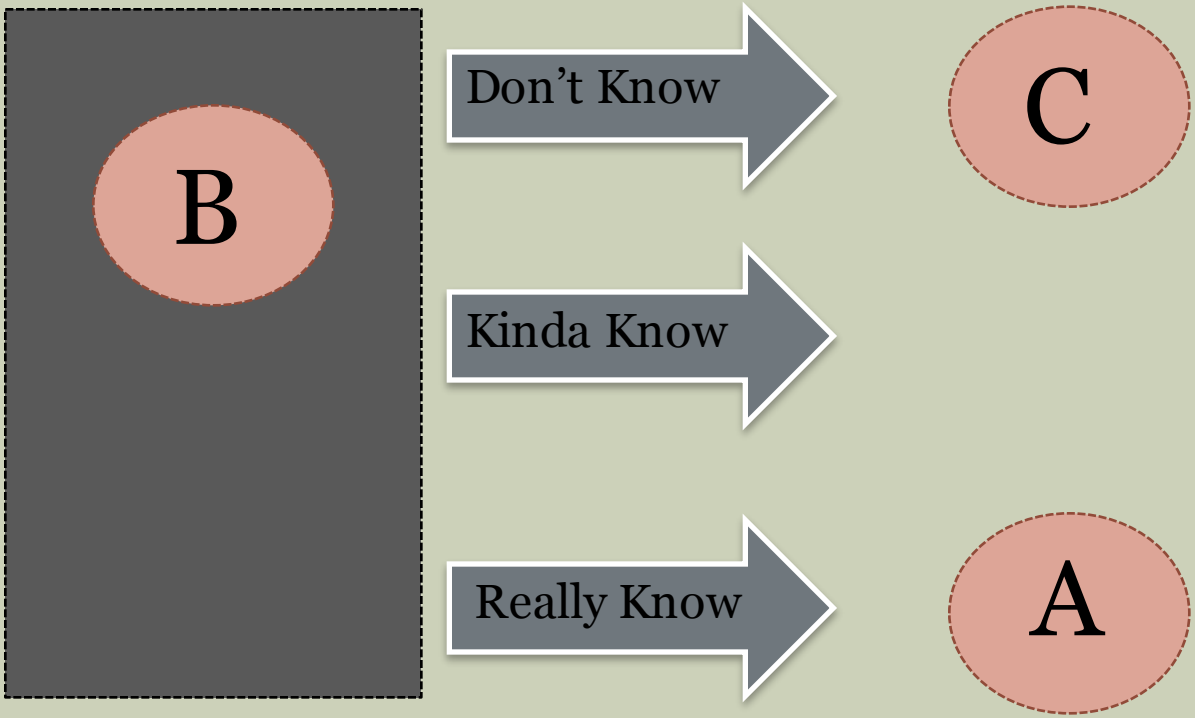
Learning Algorithm - Example

Action: User selects they kinda know B



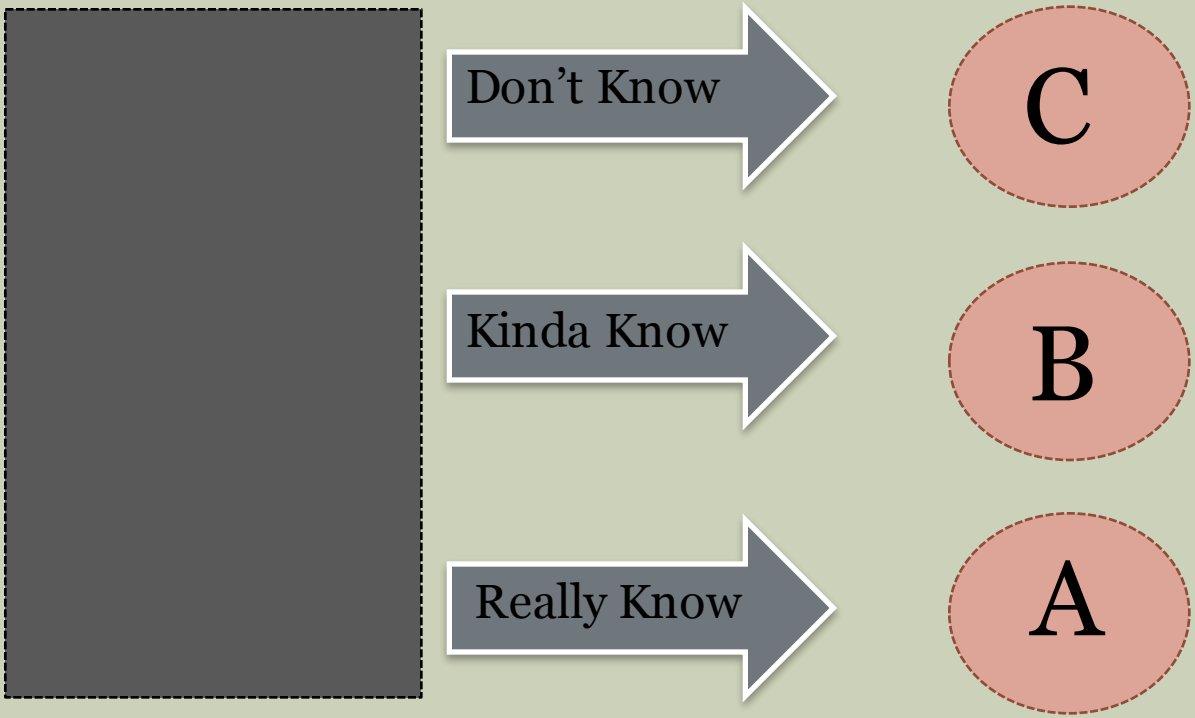
Learning Algorithm - Example

Action: B is sent to front of kinda of know queue



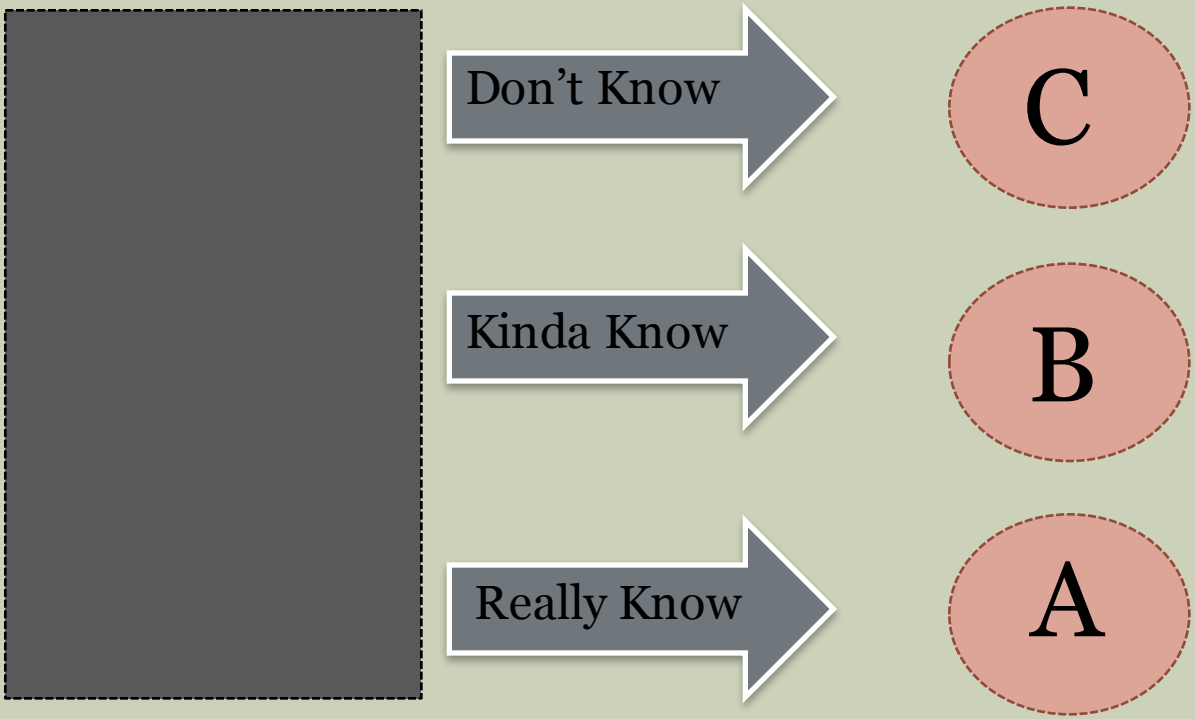
Learning Algorithm - Example

Action: B is sent to front of kinda of know queue



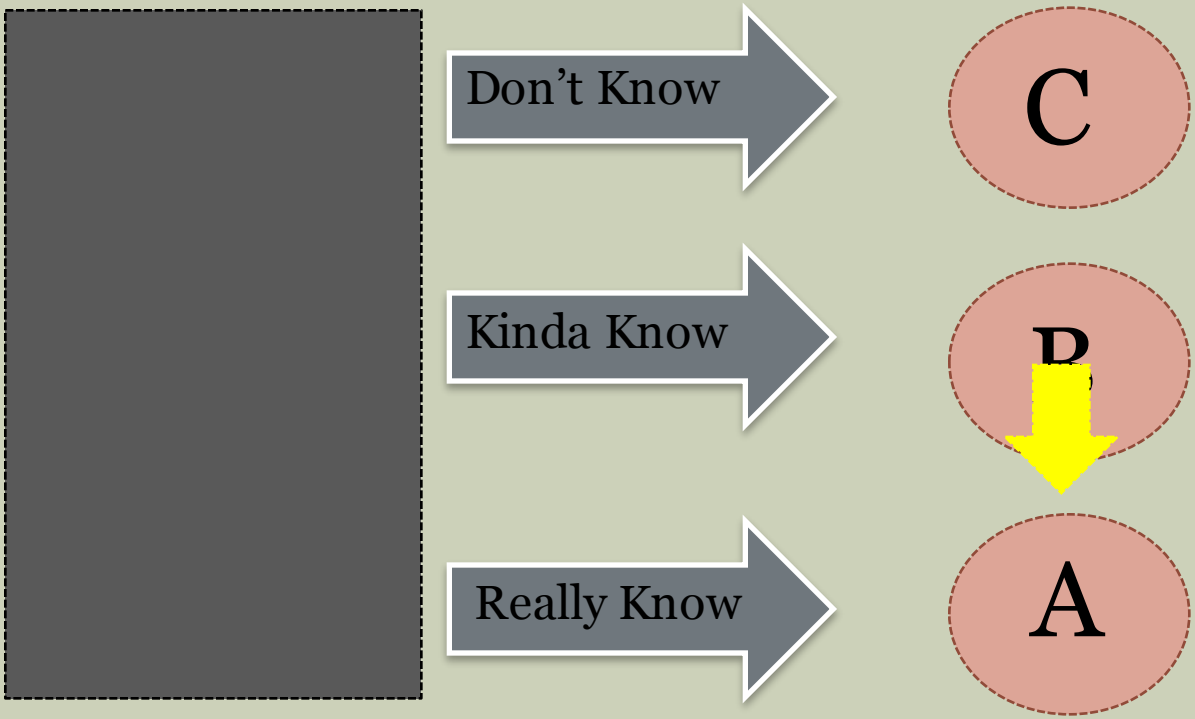
Learning Algorithm - Example

Action: Random generator generates a 6



Learning Algorithm - Example

Action: A is selected, and the algorithm continues



Cuts Made

- Beautiful user interface
  - Does not add to user-friendliness
  - Not a computer science problem
- Doodling integration
  - Impractical on a smartphone (our target device)
  - Easier to take a picture from text book/drawing

Conclusion/Advice

- Resources to Learn about Android
  - http://developer.android.com/index.html
  - http://android.programmerguru.com/simple-camera-activity
  - http://commonsware.com
- Android Apps are fun but challenging
- Many opportunities for android developers
  - Internships, Start-ups, consulting

Questions Slide

Resources

- http://developer.android.com/index.html
- http://android.programmerguru.com/simple-camera-activity
- http://commonsware.com
- http://stackoverflow.com
- http://www.dreamincode.net
- http://www.reddit.com/r/androiddev

Images Resources

- http://www.android.com
- http://media.flashcardmachine.com
- http://developer.android.com/design/patterns/app-structure.html
- http://c2499022.cdn.cloudfiles.rackspacecloud.com
- http://www.menspsychology.com
- http://www.iconfinder.com/icondetails/67495/128/camera_shutter_icon
- http://themovieblog.com/2010/08/05/in-defense-of-jim-carreys-riddler
- http://developer.android.com/reference/android/app/Activity.html

Timeline

- Installing Android SDK and ADT Plugin for Eclipse
- Research on Android development and simple demos*
- Drawing out layout of our app
- Coding and testing*
- Combining code and testing*
- Cleaning up code and fixing small bugs
- Presentation

Future Steps

- Distribution/Publishing of app
- Camera integration
- Other cool extra features (google images)

Take-away

- Fun and useful project
- Related to operating systems
- Learned a lot
- Now have Android basic experience, would like to expand in the future