Praktikum Entwicklung von Mediensystemen mit iOS

SS 2012

Prof. Dr. Michael Rohs
michael.rohs@ifi.lmu.de
MHCI Lab, LMU München

Today

- Animations
- Prototyping

Timeline

#	Date	Topic
	19.4.	Introduction & Brainstorming future mobile concepts
1	3.5.	Video watching, Introduction to iOS
	10.5.	no class (CHI Konferenz)
	17.5.	no class (Christi Himmelfahrt)
2	24.5.	More on iOS
3	31.5.	Concept finalization, paper prototyping
	7.6.	no class (Frohnleichnam)
	14.6.	Paper prototyping test, start of software prototype
5	21.6.	
6	28.6.	Think aloud study of software prototype
7	5.7.	
8	12.7.	Completion of software prototype
9	19.7.	Final presentation

ANIMATIONS

UIView Class

- Rectangular area on the screen
  - Renders content in that area
  - Handles interactions in that area
- Base class of anything visible on the screen
  - UILabel, UIButton, UIImageView, UITableView, etc.
- A view contains zero or more subviews
- Can be subclassed for custom views
  - Drawing: drawRect method, UIGraphicsGetCurrentContext
  - Trigger redrawing: setNeedsDisplay method
  - Touch events (implements UIResponder)
- Call UIView methods from the main thread only

UIView Animatable Properties

- Some properties can be gradually changed over time, i.e. animated
  - @property frame
  - @property center
  - @property transform
  - @property alpha
  - @property backgroundColor
  - (@property bounds)
  - (@property contentStretch)

UIView Animation with Blocks

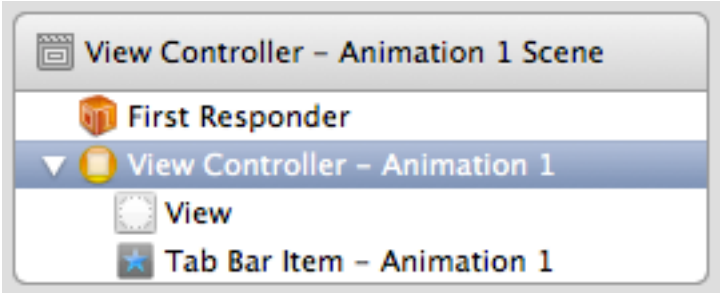
- Animate properties of one or more views over time
  - Set properties to initial values (before animation starts)
  - Set desired final values of properties
  - Optionally: Set acceleration/deceleration
  - Optional: Set operation to perform when animation is done
- Scheme:

```
view.property1 = <initial value>;
view.property2 = <initial value>;
[UIView animateWithDuration:<duration in sec>
        animations:^{
                view.property1 = <final value>;
                view.property2 = <final value>;
        }
];
```

Add Image View as Subview

- Add image view as subview when view did load
  - viewDidLoad is a method of your UIViewController subclass
  - any UIView object has zero or more subviews

```
- (void)viewDidLoad {
    [super viewDidLoad];
    UIImage *image = [UIImage imageNamed:@"testimage"];
    UIImageView *iv = [[UIImageView alloc]
                       initWithImage:image];
    [self.view addSubview:iv];
}
```



From Transparent to Opaque

- Animate view when view controller becomes active
  - Change alpha from 0 (transparent) to 1 (opaque) in 3 sec

```
- (void)viewWillAppear:(BOOL)animated {
    [super viewWillAppear:animated];
    UIImageView *iv = [[self.view subviews] objectAtIndex:0];
    iv.alpha = 0.0;
    [UIView animateWithDuration:3.0
            animations:^{
                    iv.alpha = 1.0;
            }
    ];
}
```

View Animations with Options

- Animate alpha from 0 to 1 in 1 sec; accelerate/decelerate, reverse the animation; repeat forever

```
UIImageView *iv = [[self.view subviews] objectAtIndex:0];
iv.alpha = 0.0;
[UIView animateWithDuration:1.0
        delay:0.0
        options:UIViewAnimationOptionRepeat |
                UIViewAnimationOptionCurveEaseInOut |
                UIViewAnimationOptionAutoreverse
        animations:^{
                iv.alpha = 1.0;
        }
        completion:nil];
```

Simultaneous Animation of Two Properties

- Animate multiple properties simultaneously
  - Change alpha and move center from left/top to display center in 3s

```
- (void)viewWillAppear:(BOOL)animated {
    [super viewWillAppear:animated];
    UIImageView *iv = [[self.view subviews] objectAtIndex:0];
    iv.alpha = 0.0;
    iv.center = CGPointMake(0, 0);
    CGRect frame = self.view.frame;
    [UIView animateWithDuration:3
            animations:^{
                    iv.alpha = 1.0;
                    iv.center = CGPointMake(frame.size.width / 2, frame.size.height / 2);
            }
    ];
}
```

UIView frame

- CGRect **frame**: the view's location and size in the coordinate system of its superview
  - struct CGRect { CGPoint origin; CGSize size; };
  - struct CGPoint { CGFloat x; CGFloat y; };
  - struct CGSize { CGFloat width; CGFloat height; };
  - typedef float CGFloat;
- **frame** is invalid if **transform** is not the identity transform
- Coordinates are specified in "points"
  - iPhone and iPod touch: 320 x 480 points
  - iPad: 768 x 1024 points
  - "Old" iPhones: 1 point = 1 pixel
  - "New" iPhones 1 point = 2 pixels

UIView center

- CGPoint **center**: the view's center in the coordinate system of its superview
  - struct CGPoint { CGFloat x; CGFloat y; };
- Changing center updates frame.origin
- Coordinates are specified in "points"

UIView Transforms

- view.transform property
  - Describes view's affine transformation

- Identity: CGAffineTransformIdentity
- Translation: CGAffineTransformMakeTranslation(tx, ty)
- Rotation: CGAffineTransformMakeRotation(angle)
- Scaling: CGAffineTransformMakeScale(sx, sy)

- Concatenation with current view.transform
  - CGAffineTransformRotate(CGAffineTransform t, angle)
  - CGAffineTransformScale(CGAffineTransform t, sx, sy)
  - CGAffineTransformTranslate(CGAffineTransform t, tx, ty)

UIView Transforms in Animations

- Animate multiple properties simultaneously
  - Change alpha, scale from 10% to 100%, and rotate by 90° in 3 sec

```
- (void)viewWillAppear:(BOOL)animated {
    [super viewWillAppear:animated];
    UIImageView *iv = [[self.view subviews] objectAtIndex:0];
    iv.alpha = 0.0;
    iv.transform = CGAffineTransformMakeScale(0.1, 0.1);
    [UIView animateWithDuration:3
        animations:^{
            iv.alpha = 1.0;
            iv.transform = CGAffineTransformMakeRotation(M_PI_2);
        }
    ];
}
```

Two Consecutive Animations

- Rotate 90° (3 sec), then rotate back and scale to 50%

```
- (void)viewWillAppear:(BOOL)animated {
    [super viewWillAppear:animated];
    UIImageView *iv = [[self.view subviews] objectAtIndex:0];
    iv.transform = CGAffineTransformIdentity;
    [UIView animateWithDuration:3 delay:0
        options:UIViewAnimationOptionCurveEaseInOut
        animations:^{
            iv.transform = CGAffineTransformMakeRotation(M_PI_2); }
        completion:^(BOOL finished) {
            [UIView animateWithDuration:3
                animations:^{
                    iv.transform = CGAffineTransformMakeScale(0.5, 0.5); }
            ];
    }];
}
```

New Features in iOS5

- Storyboards ✔
- Automati Reference Counting ✔
- iCloud
- Twitter integration
- UIKit additions / changes
- Newsstand
- Notification center
- Location simulation
- and more…

CORE ANIMATION LAYERS

There is more to UIViews than meets the eye…

- UIViews contain CALayers
  - Normally not a problem, but may be when animating
- UI elements not responsive during animation
  - (Partial) solution: UIViewAnimationOptionAllowUserInteraction
- Example:

```
[UIView animateWithDuration:5.0 delay:0.0
    options:UIViewAnimationOptionAllowUserInteraction |
            UIViewAnimationOptionCurveEaseInOut
    animations:^{
        label.transform = CGAffineTransformMakeTranslation(500, 0);
    }
    completion:nil];
```

UIViews contain CALayers

- Problem: At start of animation, UIView frame immediately gets end-of-animation state
  - Touch input related to end-of-animation state
- Example: translation 500 points in x direction in 5s
  - Result: Label appears to be non-responsive during animation

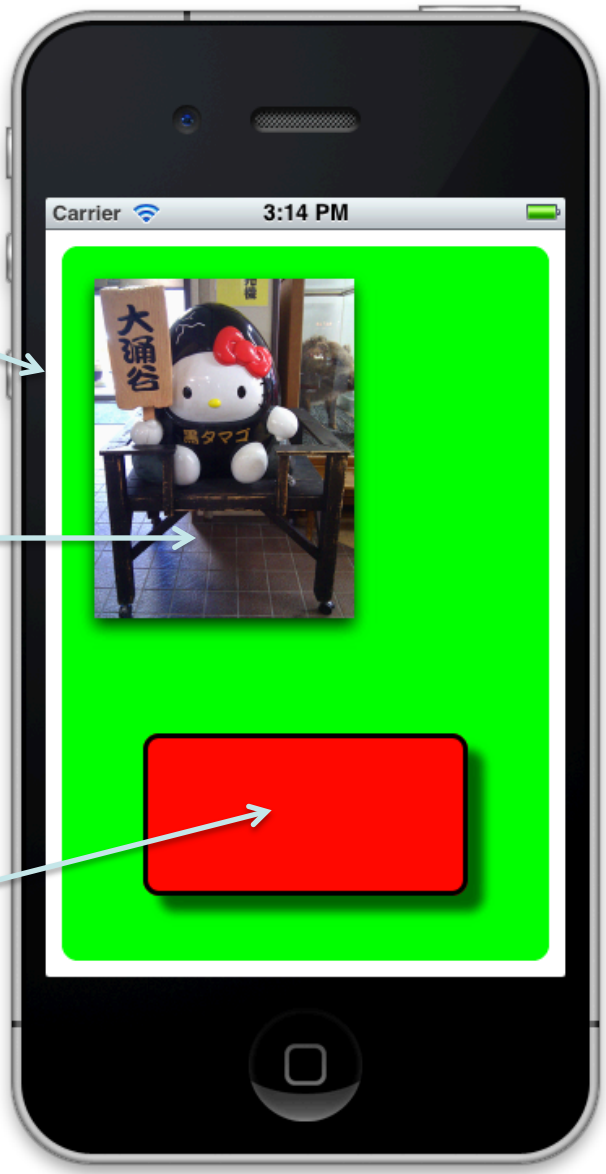
- Why?
  - UIViews contain one or more Core Animation Layers (CALayers)
  - CALayers represent a rectangular area on the screen
  - CALayers have animated properties
  - CALayers do not process user input
  - UIViews process user input → view.frame

Core Animation Layers (class CALayer)

- Represent rectangulars area on the screen
- Have animated properties
- Do not process user input (UIViews do)
  - UIView processes user input if touch point within view.frame
- UIView contains one or more CALayers
  - CALayer* l = label.layer;
- Can form hierarchies (superlayer, sublayers)
- Define geometry (position, size, transform)
- Can have content (image, Core Graphics drawing, color)
- Appearance (shadow, rounded edges, insets, etc.)
- Framwork QuartzCore, #import "QuartzCore/CALayer.h"

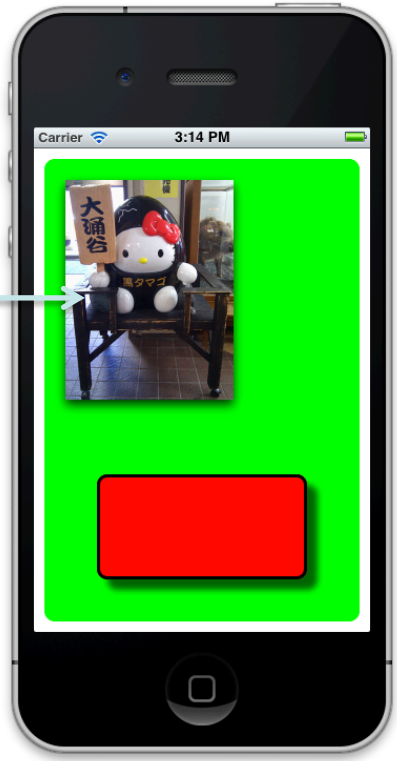
Example CALayer Hierarchy

```
self.view.layer.backgroundColor =
    [UIColor greenColor].CGColor;
self.view.layer.cornerRadius = 10.0;
self.view.layer.frame =
    CGRectInset(self.view.layer.frame, 10, 10);

CALayer *imageLayer = [CALayer layer];
imageNamed:@"testimage.png"].CGImage;
…
[self.view.layer addSublayer:imageLayer];

CALayer *sublayer = [CALayer layer];
sublayer.backgroundColor =
   [UIColor redColor].CGColor;
…
[self.view.layer addSublayer:sublayer];
```



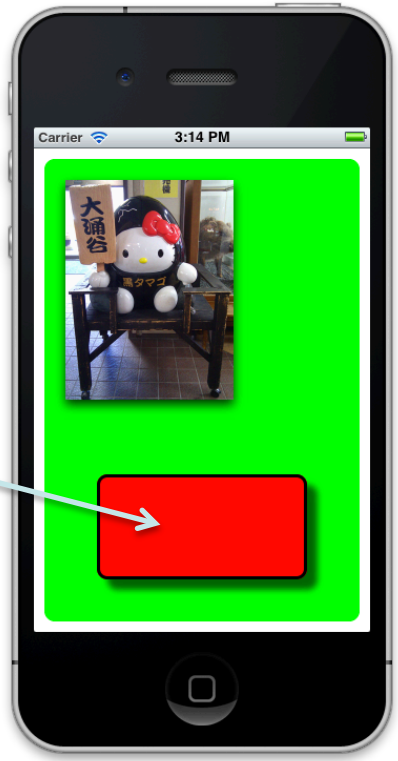
Example CALayer Hierarchy

```
CALayer *imageLayer = [CALayer layer];
imageLayer.frame = CGRectMake(20, 20, 160, 214);
imageLayer.cornerRadius = 10.0;
imageLayer.contents = (id) [UIImage
imageNamed:@"testimage.png"].CGImage;
imageLayer.shadowOffset = CGSizeMake(0, 5);
imageLayer.shadowRadius = 5.0;
imageLayer.shadowColor = [UIColor blackColor].CGColor;
imageLayer.shadowOpacity = 0.8;
[self.view.layer addSublayer:imageLayer];
```



Example CALayer Hierarchy

```
CALayer *sublayer = [CALayer layer];
sublayer.backgroundColor = [UIColor redColor].CGColor;
sublayer.shadowOffset = CGSizeMake(10, 10);
sublayer.shadowRadius = 3.0;
sublayer.shadowColor = [UIColor blackColor].CGColor;
sublayer.shadowOpacity = 0.6;
sublayer.frame = CGRectMake(50, 300, 200, 100);
sublayer.borderColor = [UIColor blackColor].CGColor;
sublayer.borderWidth = 3.0;
sublayer.cornerRadius = 10.0;
[self.view.layer addSublayer:sublayer];
```



Core Animation Rendering Architecture

- CALayers encapsulate geometry, timing, visual properties
- Visible layer tree is backed by presentation and render trees
  - Layer tree: property values (model)
  - Presentation tree: state that is currently shown, gradually changes during animation

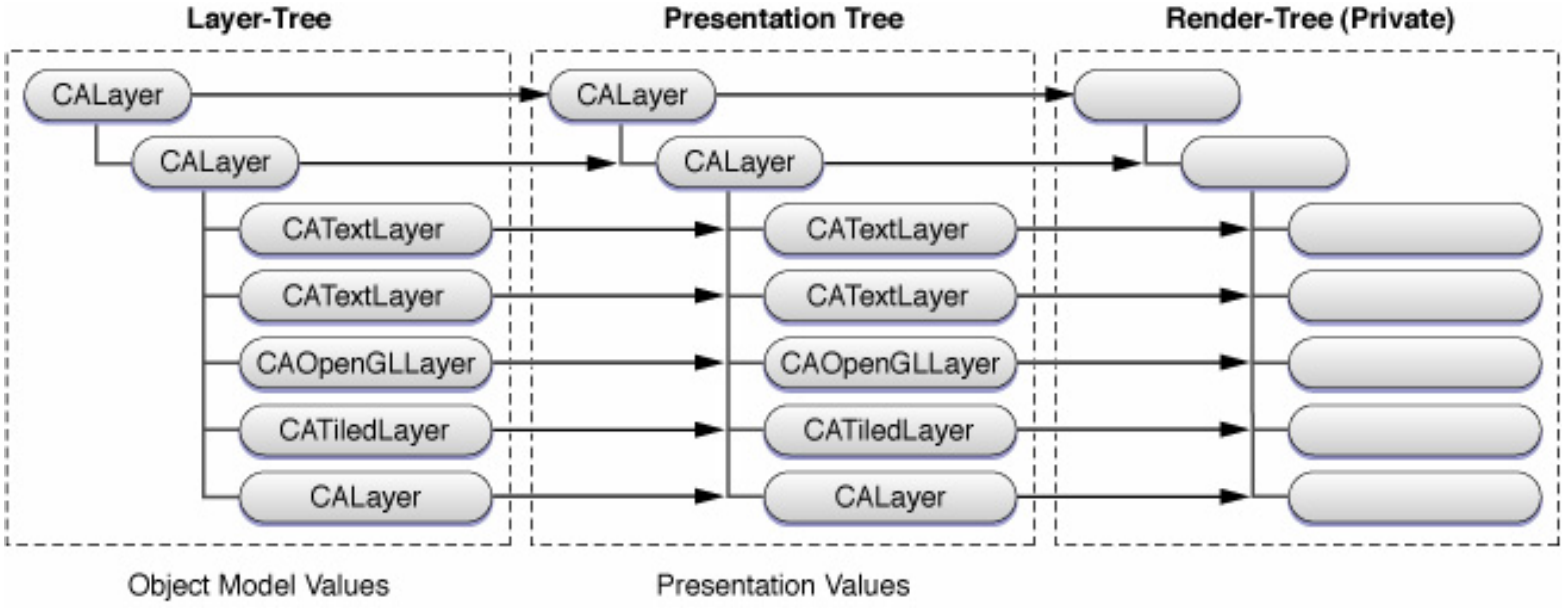


Image source: Apple documentation

CALayer Geometry and Transforms

- Each layer has its own coordinate system
  - Position, frame, bounds, anchor point
- Transform as known from UIViews

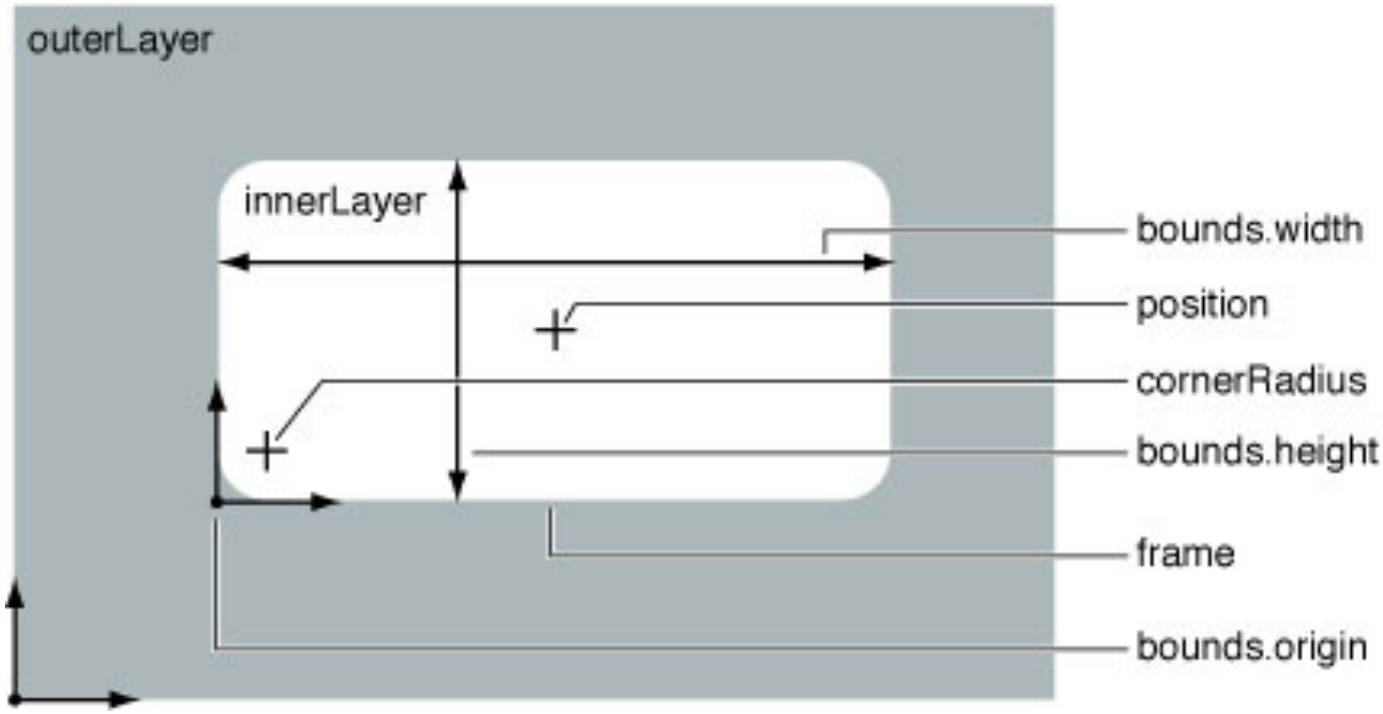


Image source: Apple documentation

Resources on Core Animation

- http://developer.apple.com/library/ios/#documentation/Cocoa/Conceptual/CoreAnimation_guide
- http://www.raywenderlich.com/2502/introduction-to-calayers-tutorial
- http://www.raywenderlich.com/2454/how-to-use-uiview-animation-tutorial

Solution: Derive from UIView-Subclass

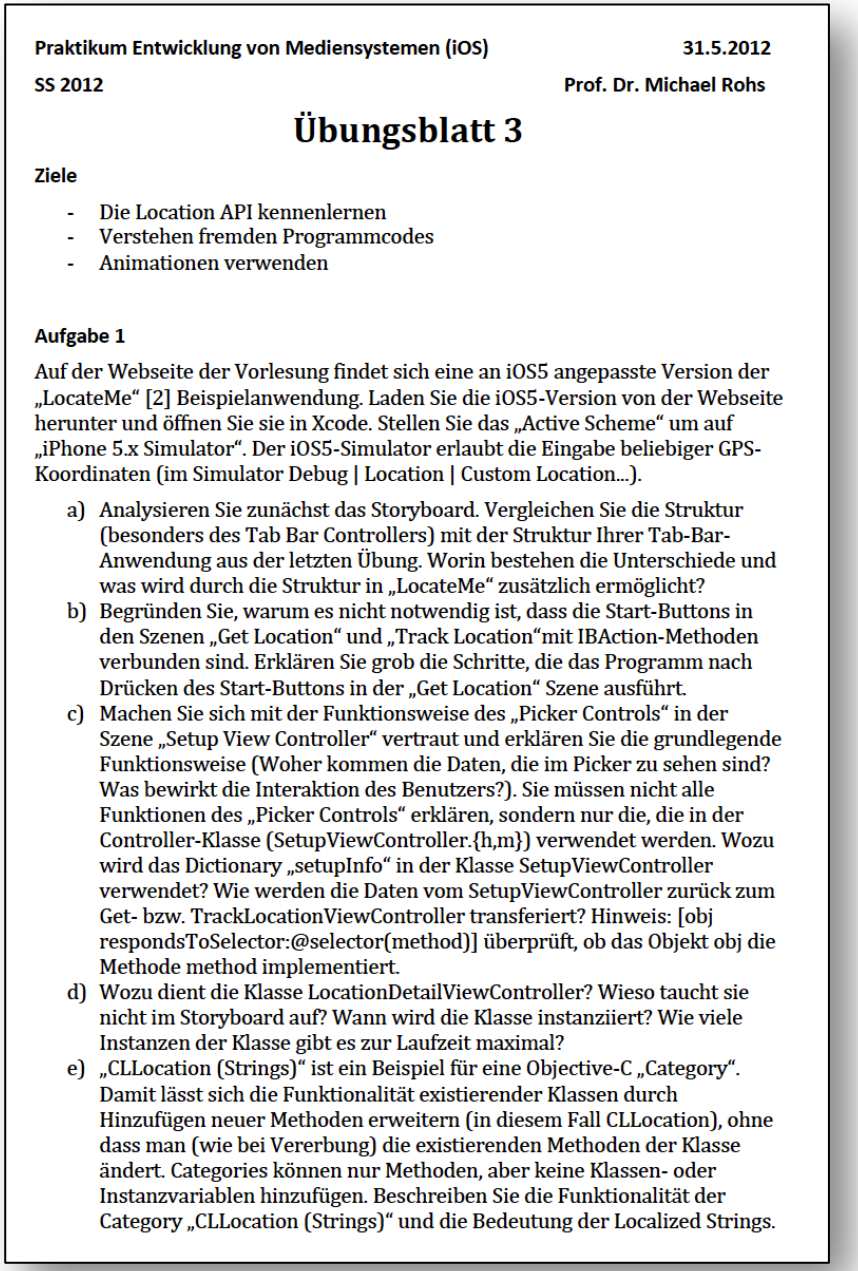
- Implement hit test your self

```
- (UIView *)hitTest:(CGPoint)point withEvent:(UIEvent *)event {
        UIView *v = [super hitTest:point withEvent:event];
        return v;
}

- (BOOL)pointInside:(CGPoint)point withEvent:(UIEvent *)event {
        CALayer *modelLayer = [[self layer] modelLayer];
        CALayer *presentationLayer = [[self layer] presentationLayer];
        CGPoint point2 = [presentationLayer convertPoint:point
                                               fromLayer:modelLayer];
        BOOL b = [presentationLayer containsPoint:point2];
        return b;
}
```

AUFGABE

Exercise 3

- Ziele
  - Die Location API kennenlernen
  - Verstehen fremden Programmcodes
  - Animationen verwenden



Praktikum Entwicklung von Mediensystemen (iOS) 31.5.2012

SS 2012 Prof. Dr. Michael Rohs

Übungsblatt 3

Ziele

- Die Location API kennenlernen
- Verstehen fremden Programmcodes
- Animationen verwenden

Aufgabe 1

Auf der Webseite der Vorlesung findet sich eine an iOS5 angepasste Version der „LocateMe“ [2] Beispielanwendung. Laden Sie die iOS5-Version von der Webseite herunter und öffnen Sie sie in Xcode. Stellen Sie das „Active Scheme“ um auf „iPhone 5.x Simulator“. Der iOS5-Simulator erlaubt die Eingabe beliebiger GPS-Koordinaten (im Simulator Debug | Location | Custom Location...).

a) Analysieren Sie zunächst das Storyboard. Vergleichen Sie die Struktur (besonders des Tab Bar Controllers) mit der Struktur Ihrer Tab-Bar-Anwendung aus der letzten Übung. Worin bestehen die Unterschiede und was wird durch die Struktur in „LocateMe“ zusätzlich ermöglicht?

b) Begründen Sie, warum es nicht notwendig ist, dass die Start-Buttons in den Szenen „Get Location“ und „Track Location“mit IBAction-Methoden verbunden sind. Erklären Sie grob die Schritte, die das Programm nach Drücken des Start-Buttons in der „Get Location“ Szene ausführt.

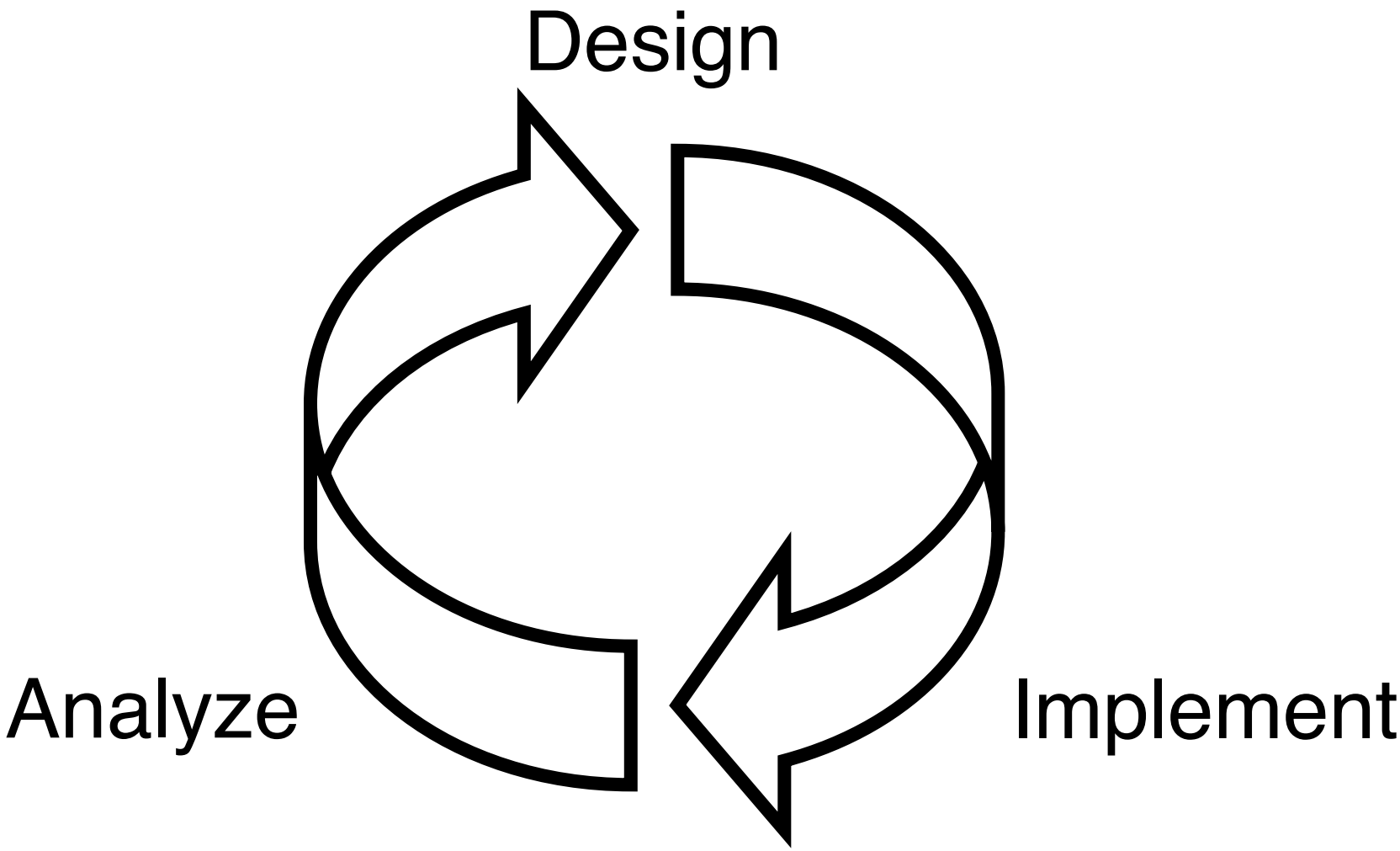
c) Machen Sie sich mit der Funktionsweise des „Picker Controls“ in der Szene „Setup View Controller“ vertraut und erklären Sie die grundlegende Funktionsweise (Woher kommen die Daten, die im Picker zu sehen sind? Was bewirkt die Interaktion des Benutzers?). Sie müssen nicht alle Funktionen des „Picker Controls“ erklären, sondern nur die, die in der Controller-Klasse (SetupViewController.{h,m}) verwendet werden. Wozu wird das Dictionary „setupInfo“ in der Klasse SetupViewController verwendet? Wie werden die Daten vom SetupViewController zurück zum Get- bzw. TrackLocationViewController transferiert? Hinweis: [obj respondsToSelector:@selector(method)] überprüft, ob das Objekt obj die Methode method implementiert.

d) Wozu dient die Klasse LocationDetailViewController? Wieso taucht sie nicht im Storyboard auf? Wann wird die Klasse instanziiert? Wie viele Instanzen der Klasse gibt es zur Laufzeit maximal?

e) „CLLocation (Strings)“ ist ein Beispiel für eine Objective-C „Category“. Damit lässt sich die Funktionalität existierender Klassen durch Hinzufügen neuer Methoden erweitern (in diesem Fall CLLocation), ohne dass man (wie bei Vererbung) die existierenden Methoden der Klasse ändert. Categories können nur Methoden, aber keine Klassen- oder Instanzvariablen hinzufügen. Beschreiben Sie die Funktionalität der Category „CLLocation (Strings)“ und die Bedeutung der Localized Strings.

DESIGN PROCESS

Iterative Design: DIA Cycle



Focus on Users

- Decide **who** the users will be
- Decide **what** they will be doing with the system

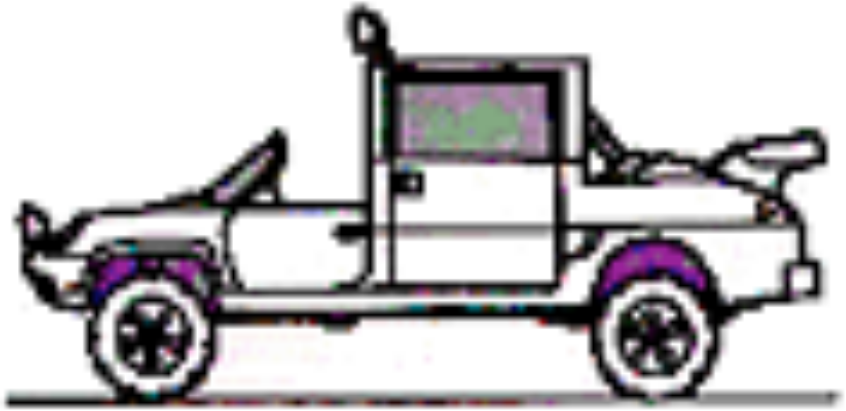
- "You can't figure out what people want, need, can do, and will do without talking to them."
- Find real people interested in your planned system (otherwise there's a problem)
- Methods
  - Talk with users
  - Visit user locations, observe (and videotape) users working
  - Have users think aloud, try it yourself
  - Use surveys and questionnaires

User Profiles or “Personas” (Cooper, 1998)

- Short profiles of typical users
  - Prototypical user for a specific user group
  - Fictitious individual with concrete characteristics
- Building personas
  - Often built from interview results
  - Synthesize fictitious users from real user characteristics
  - Develop multiple personas for different types of users
- Bring them to life
  - With a name, characteristics, experience, personal background, environment they are located in, goals, tasks, skill levels, etc.
- Base design decisions on the needs of the personas

Personas Example
(Cooper, About Face, Chapter 5)

Building a car that pleases everyone



Building a car based on three personas (representing larger groups)

Marge, *mother of three*

Marge wants safety and room for many passengers. A minivan meets her needs.

Jim, *construction worker*

Jim wants cargo space and the ability to carry heavy load. A pickup truck meets his needs.

Alesandro, *software engineer*

Alesandro wants sporty looks and speed. A two-door sports car meets his needs.

Example Persona: Bob

Bob is 52 years old and works as a mechanic with an organisation offering road service to customers when their car breaks down. He has worked in the job for the past 12 years and knows it well. Many of the younger mechanics ask Bob for advice when they meet up in the depot as he always knows the answer to tricky mechanical problems. Bob likes sharing his knowledge with the younger guys, as it makes him feel a valued part of the team.

Bob works rolling day and night shifts and spends his shifts attending breakdowns and lockouts (when customers lock their keys in the car). About 20% of the jobs he attends are complex and he occasionally needs to refer to his standard issue manuals. Bob tries to avoid using the manuals in front of customers as he thinks it gives the impression he doesn't know what he's doing.

Bob has seen many changes over the years with the company and has tried his best to move with the times. However he found it a bit daunting when a new computer was installed in his van several years ago, and now he has heard rumors that the computer is going to be updated to one with a bigger screen that's meant to be faster and better.

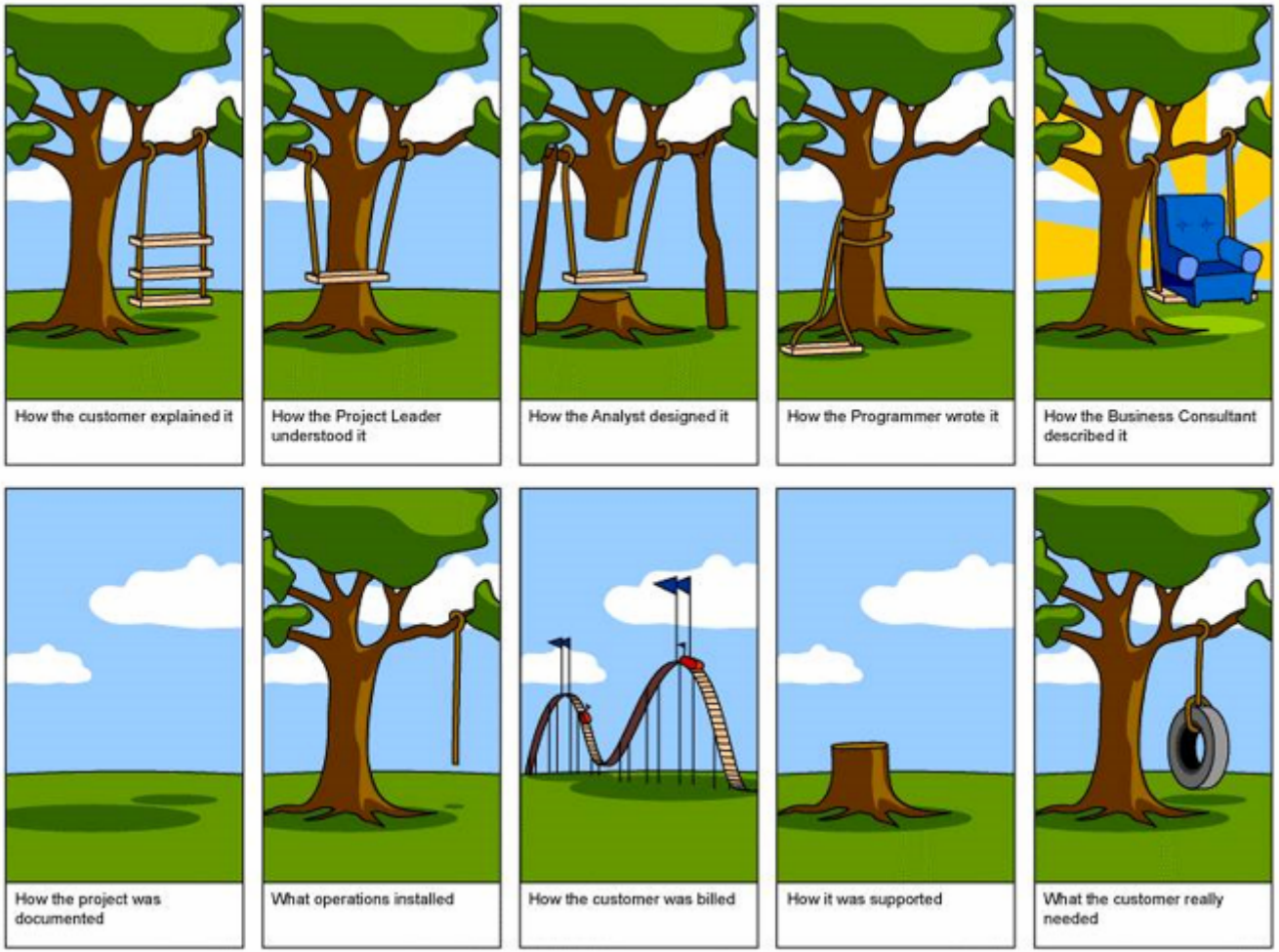
Bob's been told that he will be able to access the intranet on the new computer. He has heard about the intranet and saw once in an early version on his manager's computer. He wonders if he will be able to find out want's going on in the company more easily, especially as customers' seem to know more about the latest company news than he does when he turns up at a job. This can be embarrassing and has been a source of frustration for Bob throughout his time with the company.

Bob wonders if he will be able to cope with the new computer system. He doesn't mind asking his grandchildren for help when he wants to send an email to his brother overseas, but asking the guys at work for help is another story.

Source: http://www.steptwo.com.au/papers/kmc_personas/

Getting the Requirements Right

Major cause of project failure: unclear requirements

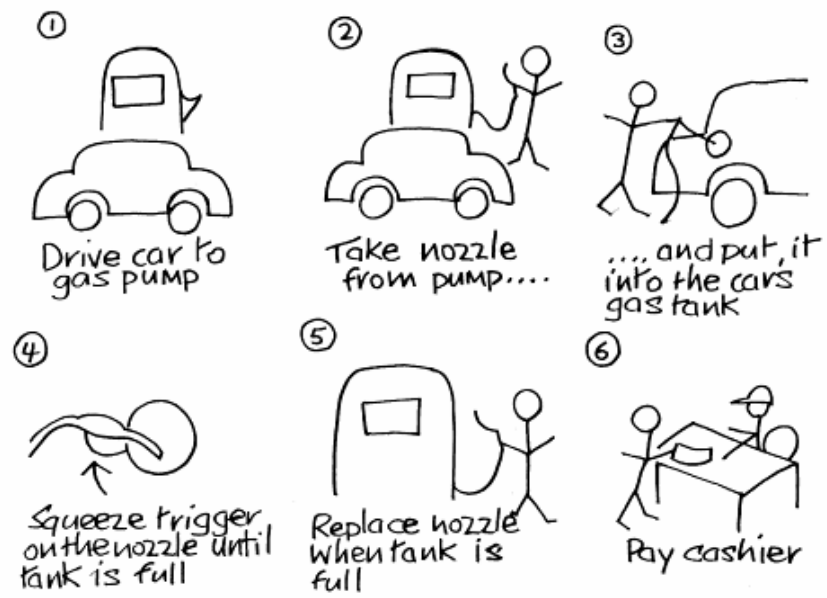


Source: Preece et al.: Interaction Design

Gathering Data

- Researching similar products
  - State-of-the-Art
  - Sets level of user expectation
- Interviews
  - Good for exploring issues
  - New perspectives
  - Props, e.g. sample scenarios, paper prototypes
- Focus groups
  - Group interviews
  - Multiple viewpoints, highlighting areas of conflict
  - Can be dominated by individuals

Initial Design Techniques: Storyboarding



- What?
  - Sequence of single images
  - Like visual outline of a movie
  - Illustrates interaction
- Why?
  - Describes task showing environment, user, and computer
  - Or describes UI as series of screen images
  - Helps working out interaction details
  - Great at-a-glance overview of interaction
  - Helps developing usage scenarios
- When?
  - After describing a task, storyboard it, then take back to user

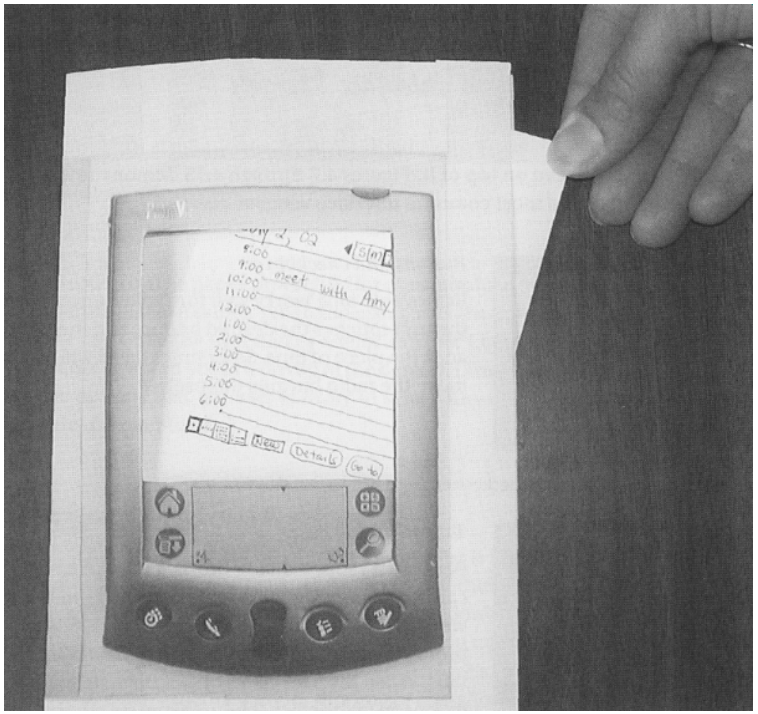
Interviews

- Unstructured
  - Not directed by a script
  - Rich but not replicable
- Structured
  - Tightly scripted, often like a questionnaire
  - Replicable but may lack richness
- Semi-structured
  - Guided by a script but free to explore interesting issues in more depth
  - Good balance between richness and replicability

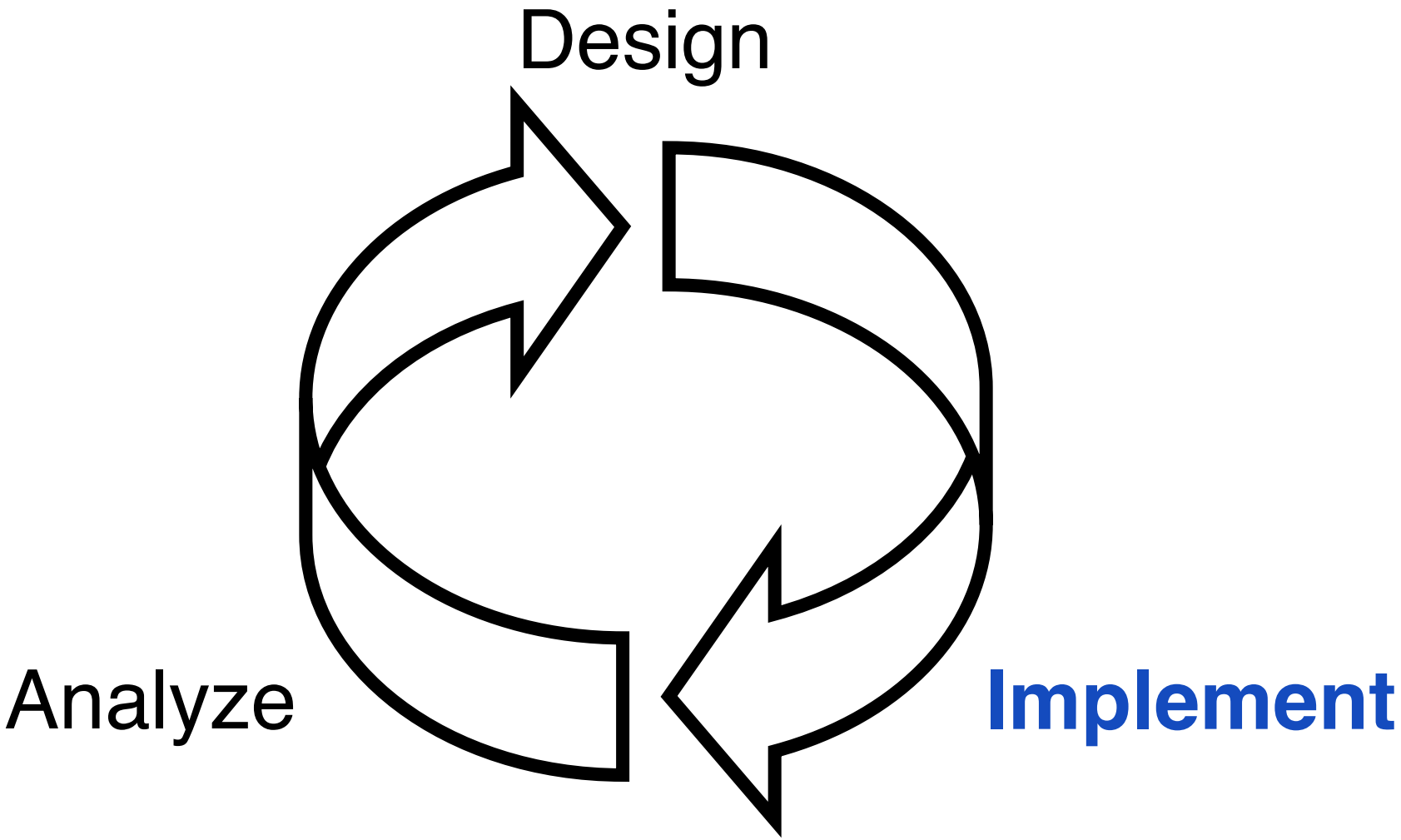
How to Ask Questions

- Clear and simple, not too broad
  - “How do you like the UI?” is too general!
- Affording logical, quantitative answers
  - Bad questions give unusable or wrong answers
  - Open vs. closed questions
- Users don’t always answer truthfully
  - Lack of knowledge, bad estimates, embarrassment
  - So formulate questions carefully, maybe indirectly
- No leading questions!
  - For initial input, do not focus on presenting your design ideas, but on learning about the task

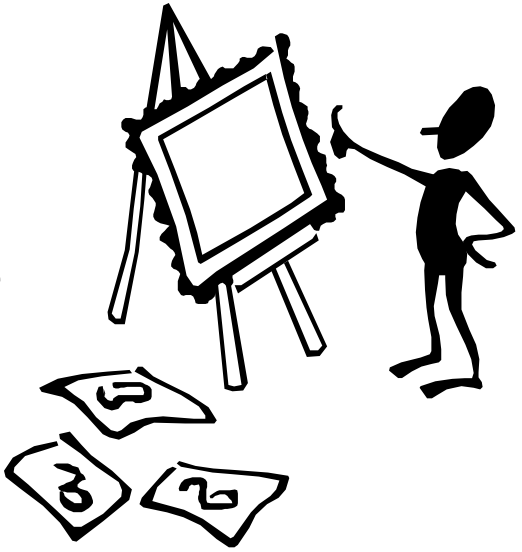
PROTOTYPING



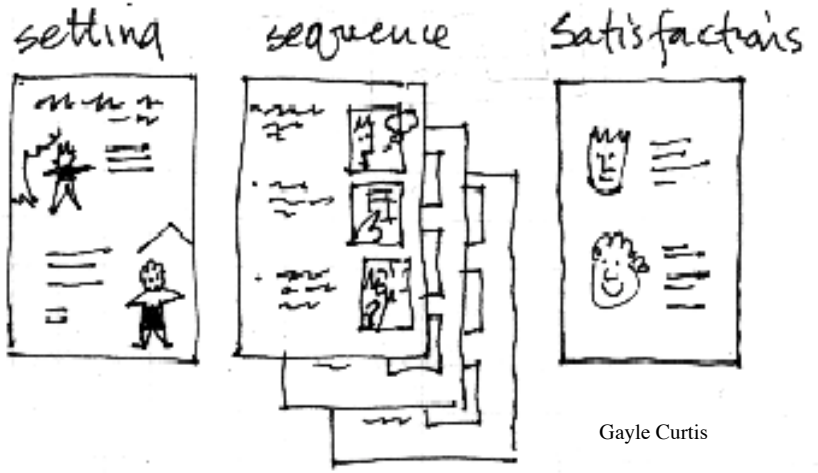
DIA Cycle: How to realize design ideas?



Low-Fidelity Paper Prototypes



- First prototype, quick and cheap
- Paper and pencil mockup of user interface
  - Rough sketches of the main screens and dialogs
  - Textual description of interface functions and relationships between screens
- Goals
  - Brainstorming
  - Expert review of interaction flow
  - First user feedback
  - User tests



Gayle Curtis

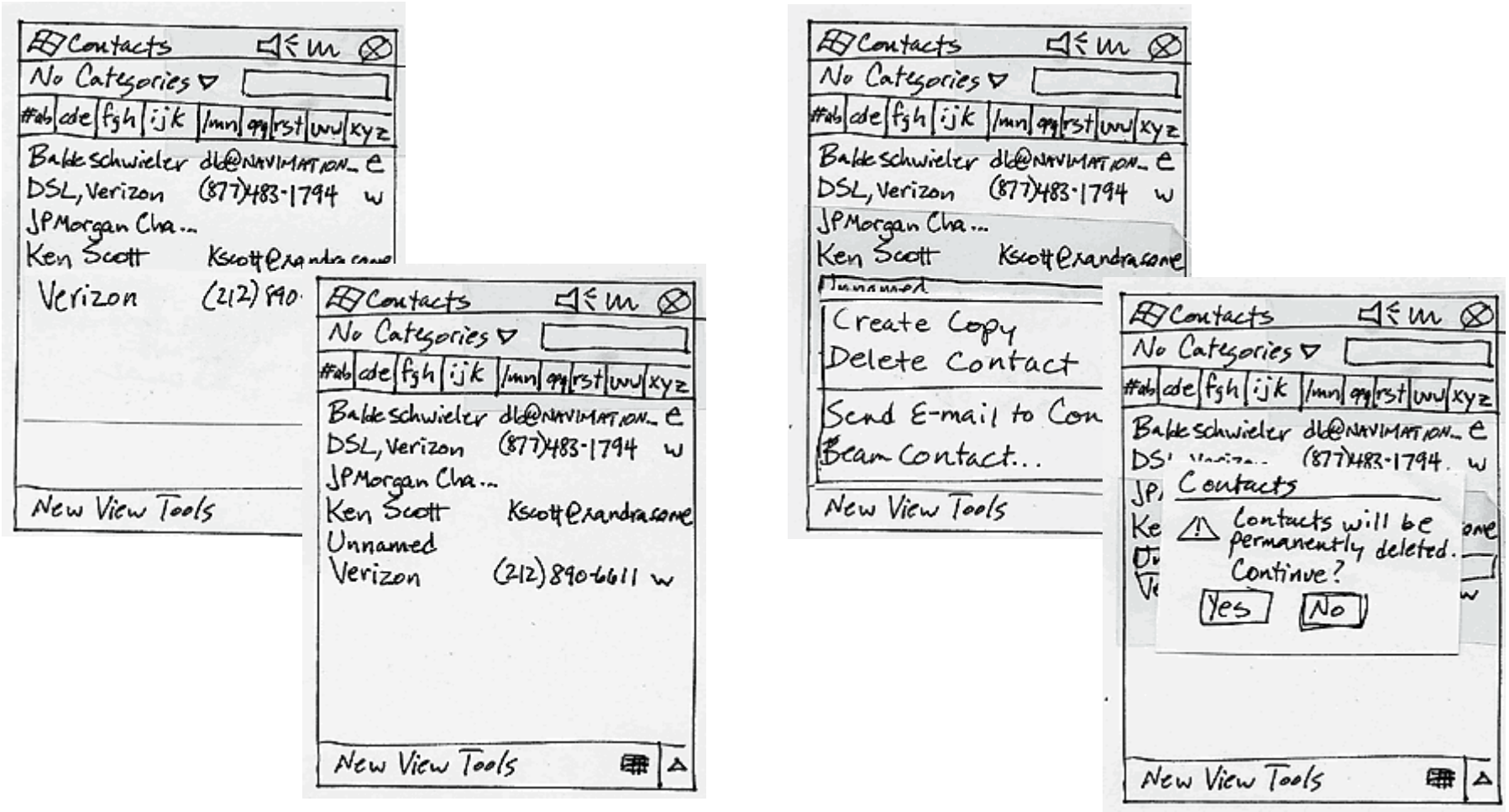
Paper / Post-it Prototype Process

Collaboratively creating the prototypes

Reviewing the prototypes

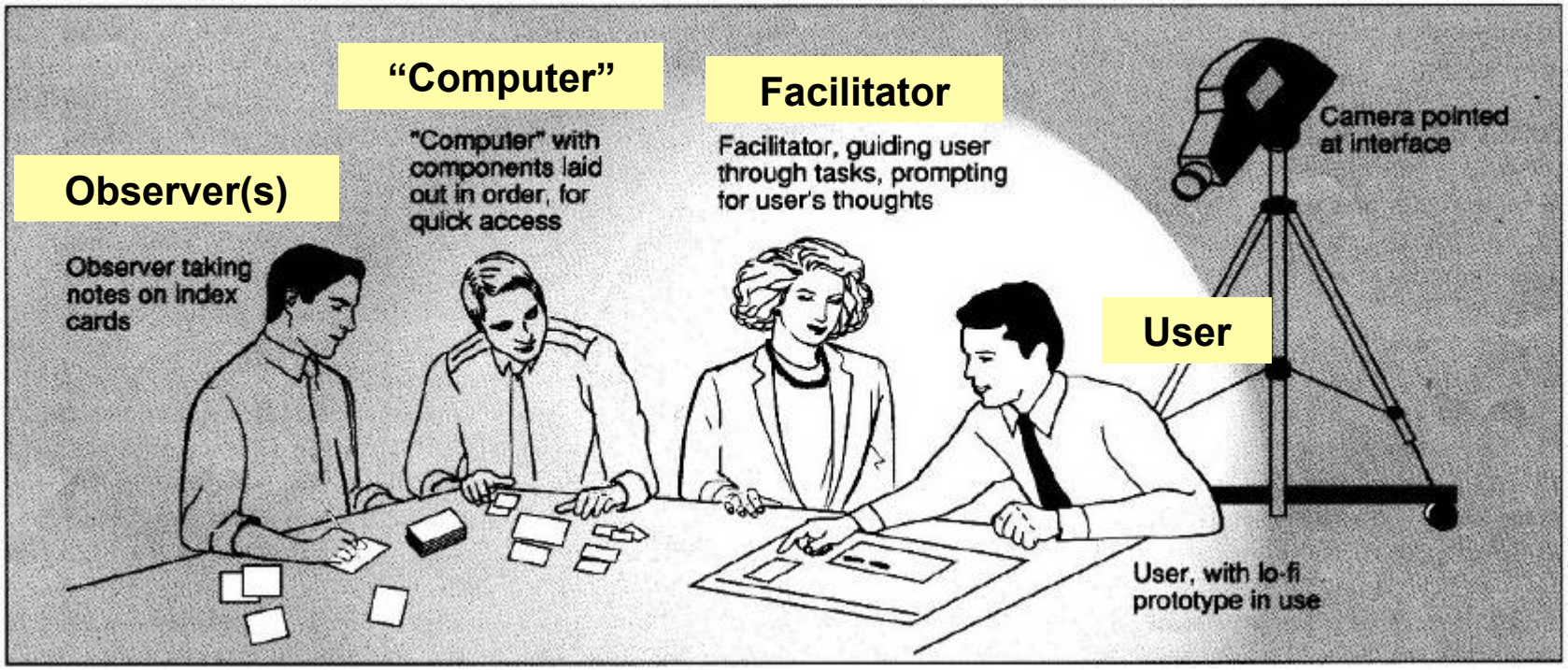
Source: http://www.pocketpcmag.com/_archives/may03/e_prototyping.asp

Paper Prototype Examples



Source: http://www.pocketpcmag.com/_archives/may03/e_prototyping.asp

Low-Fidelity User Testing

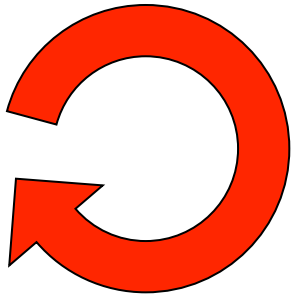


Marc Rettig: Prototyping for Tiny Fingers

- Select users
- Prepare test scenarios, drawn from task analysis
  - familiar data, realistic tasks
- Practice
  - team members know their roles, no “computer” delays

Low-Fidelity Prototype Revision

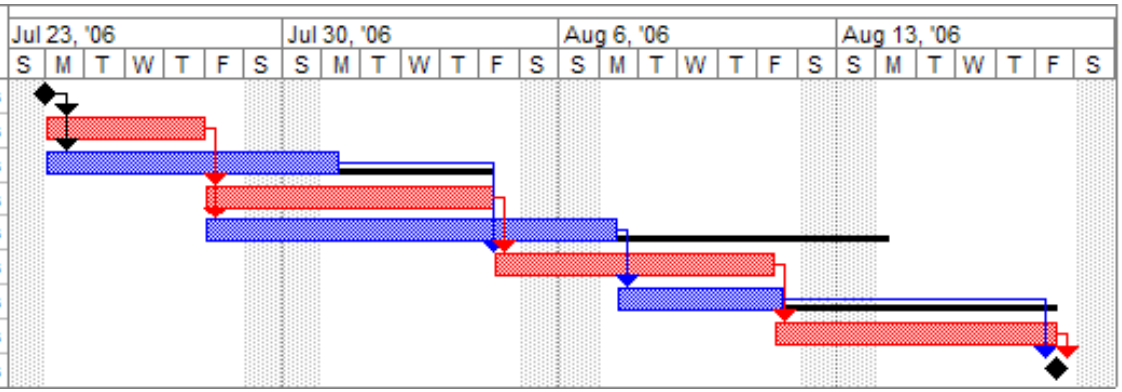
- Evaluation of test results
  - Arrange paper prototype on table
  - Pile note cards next to component
- Summarize and prioritize problems
  - Written report on findings
- Prototype refinement
  - Agenda for meeting to discuss design changes
  - Attach post-it notes with changes to each component



Recommendations

- Set up a Web site or Wiki to document the progress of your project
- Plan the communication within your team
- Create a Gantt chart to plan your project
  - en.wikipedia.org/wiki/Gantt
  - Gantt chart template on course Web page

ID	Task Name	Predecessors	Duration
1	Start		0 days
2	a	1	4 days
3	b	1	5.33 days
4	c	2	5.17 days
5	d	2	6.33 days
6	e	3,4	5.17 days
7	f	5	4.5 days
8	g	6	5.17 days
9	Finish	7,8	0 days



AUFGABE

Aufgabe

- Erstellen Sie einen Papier-Prototyp, der die einzelnen Screens/UI-Zustände und Übergänge dazwischen darstellt.
- Führen Sie einen Benutzer-Test mit dem Papier-Prototyp mit 3 Benutzern (nicht aus dem PEM-Praktikum) durch
  - Typische Aufgabe festlegen
  - Benutzer bitten, diese Aufgabe mit dem Papier-Prototyp durchzuführen
  - Protokollieren, Probleme finden
  - Papier-Prototyp verbessern
- Schreiben Sie die Erfahrungen beim Test auf
  - Was ließ sich gut testen, was war problematisch