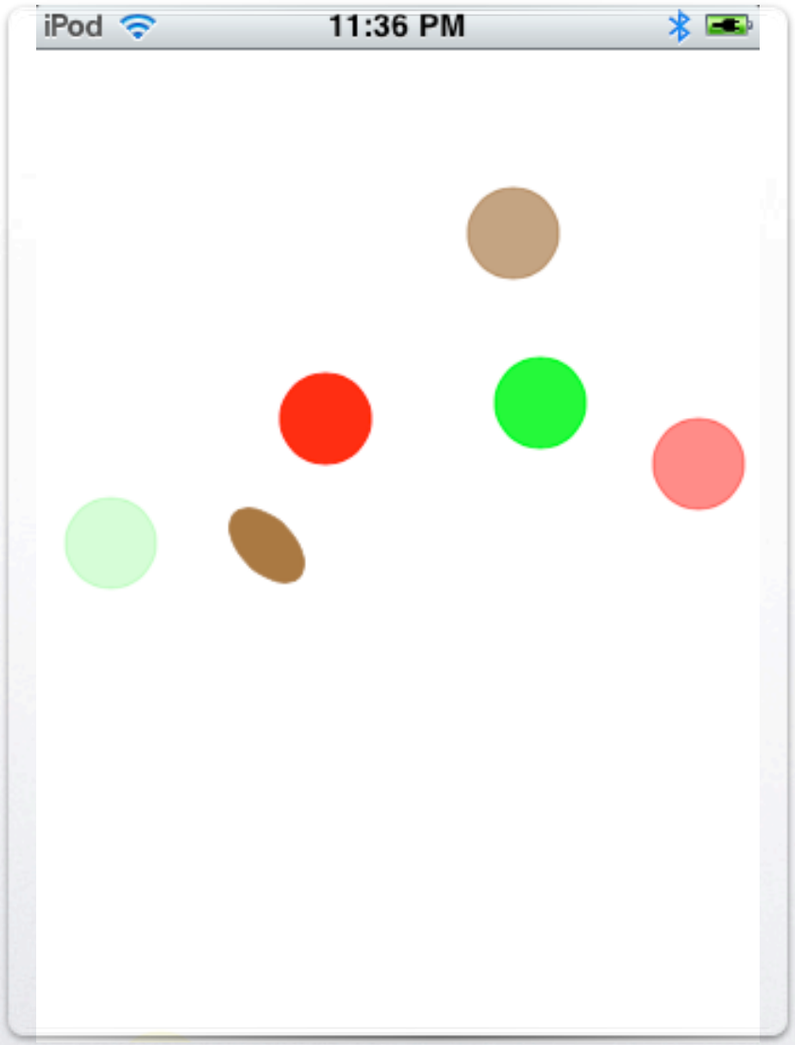
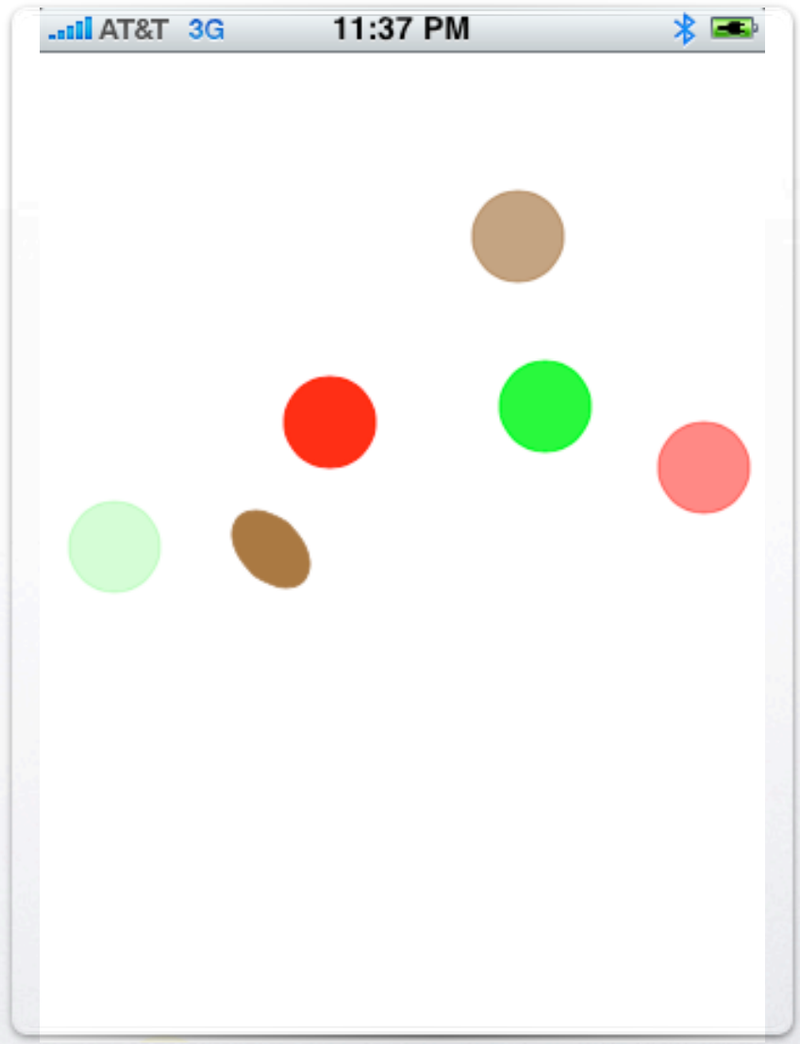
iPhone to iPhone

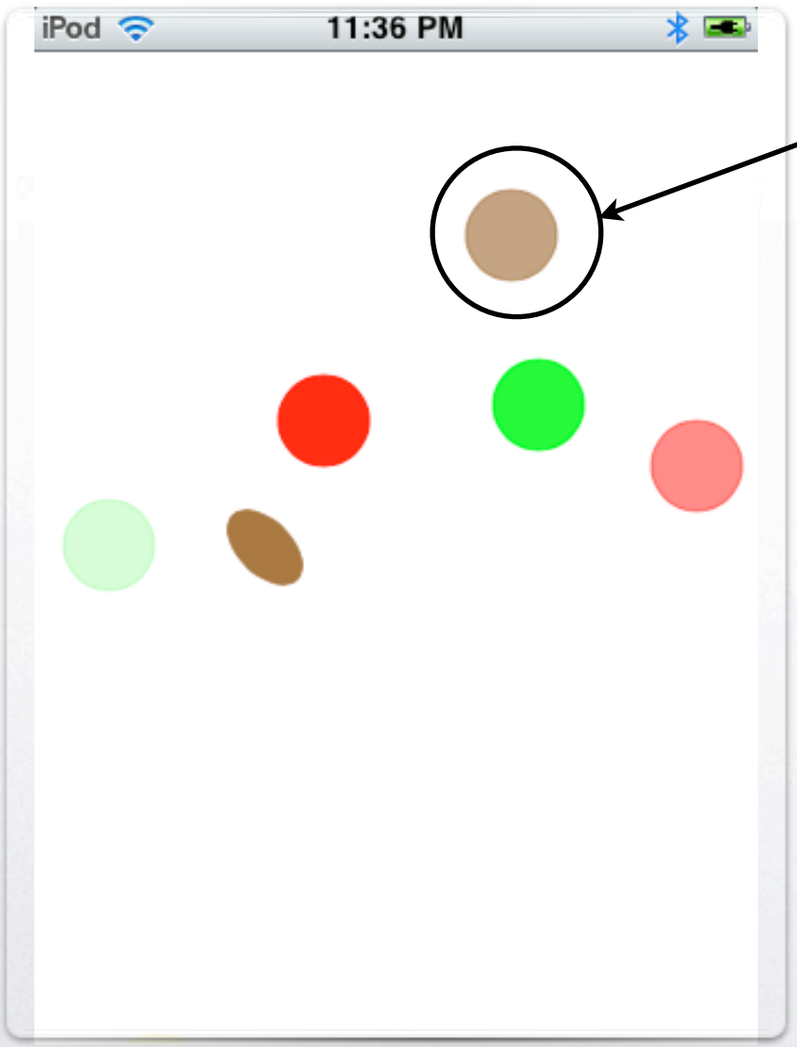
Peer to Peer networking with Game Kit

Where We Are Headed



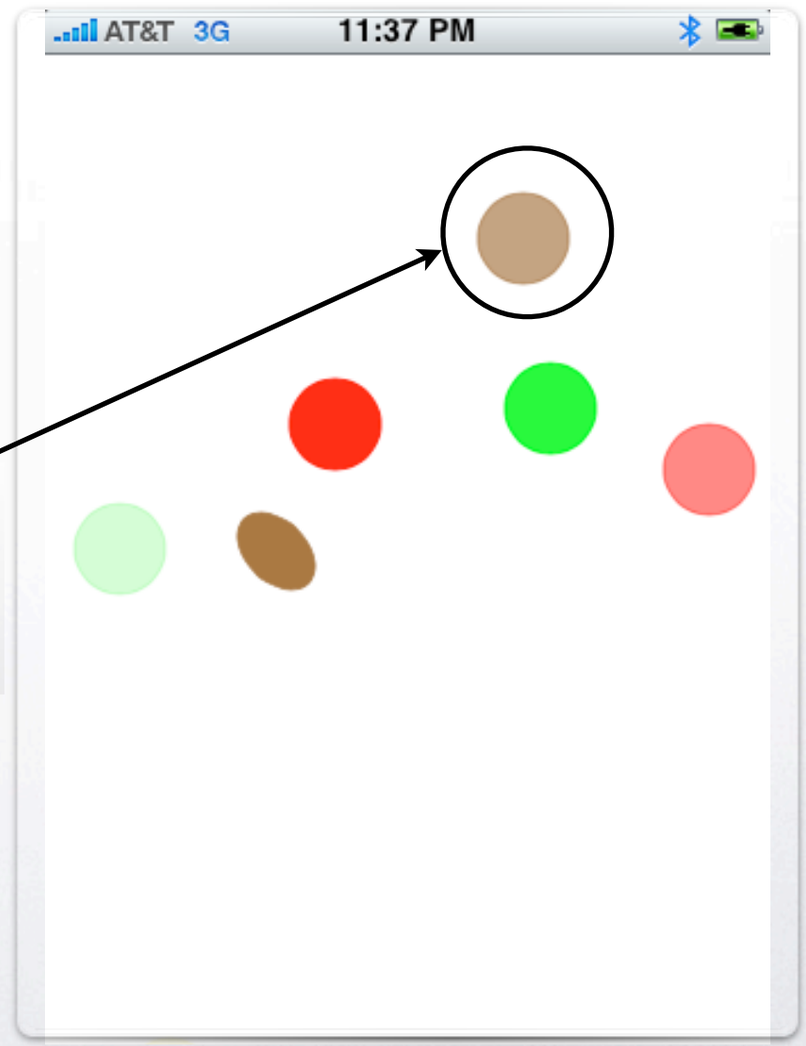


The ‘Game’

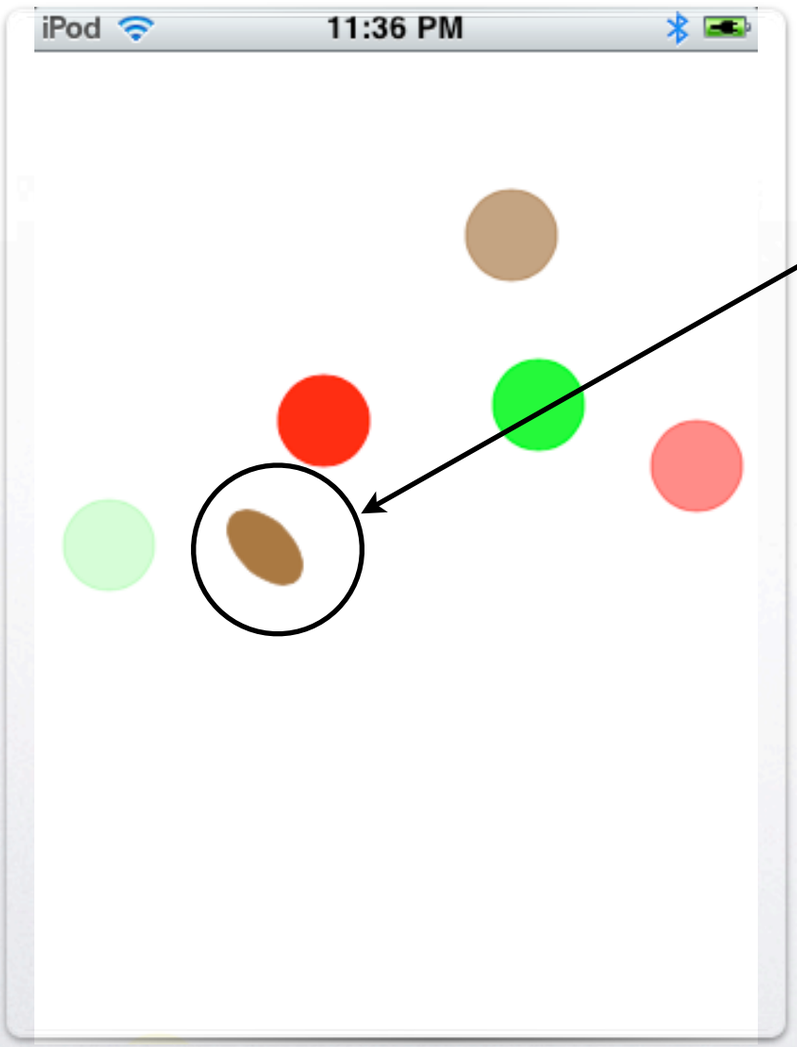


One becomes the ‘server’
and creates new disks
adds them locally to the view
and packages them up and sends
them to the client

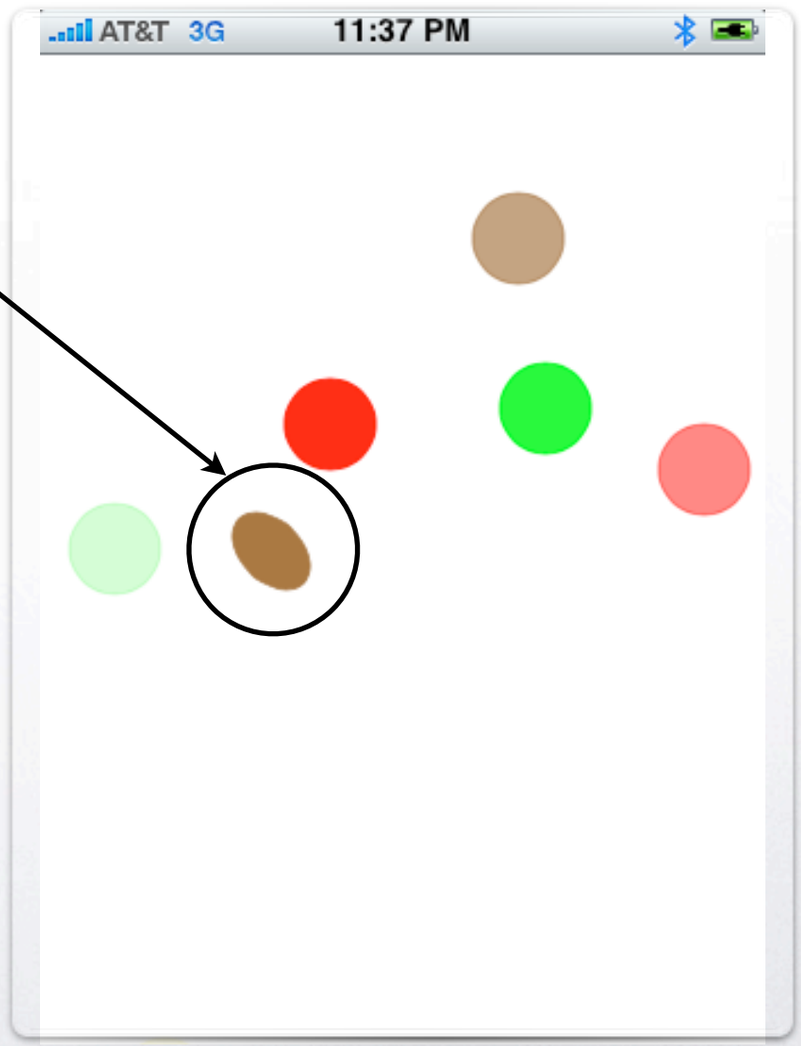
One becomes the ‘client’
listens for new disks, when found
creates a new one and adds it locally



The ‘Game’



User taps on the disks and they spin in 3D and disappear the game sends what was hit to the other peer so both know what has been hit



How to get there

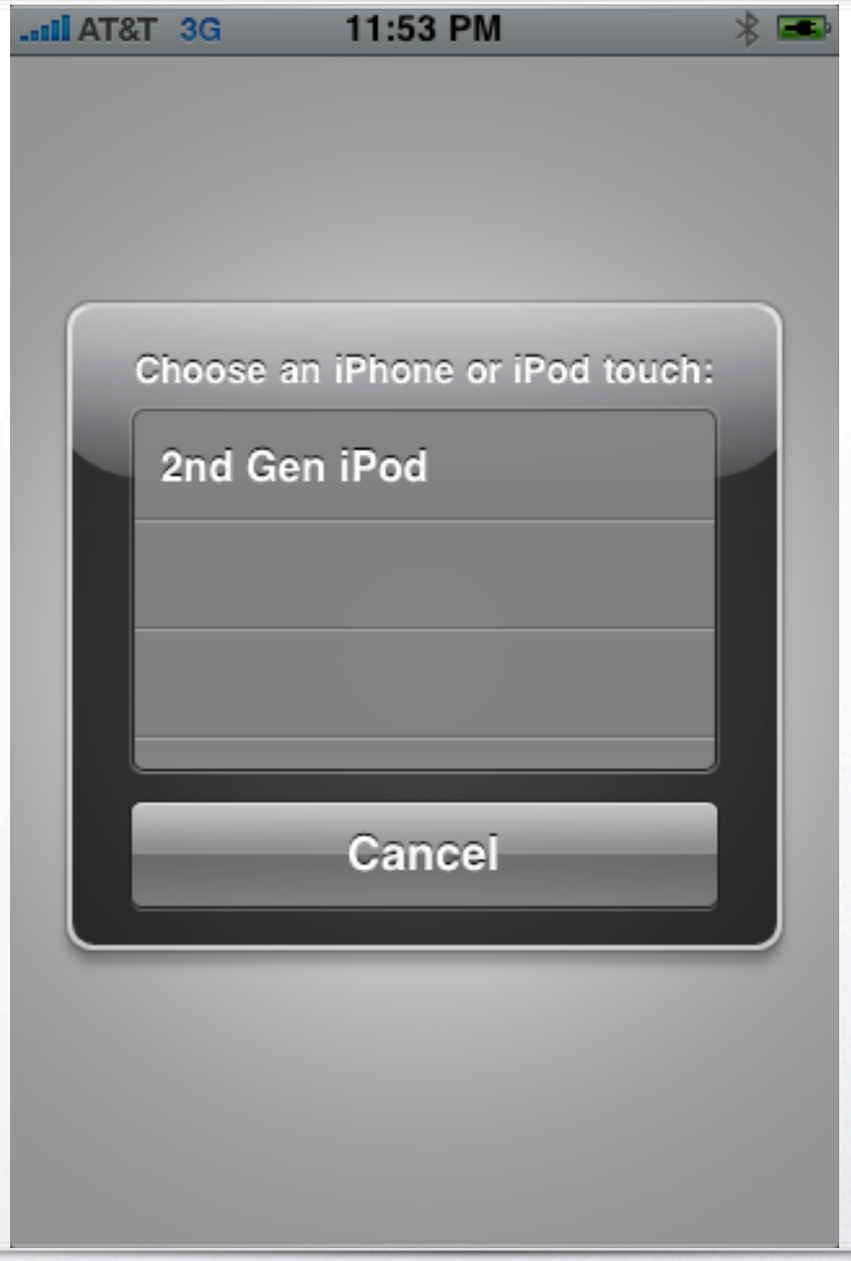
- Connect Peers
- Run Game Loop
- Process Events

Application Startup

- Start looking for peers

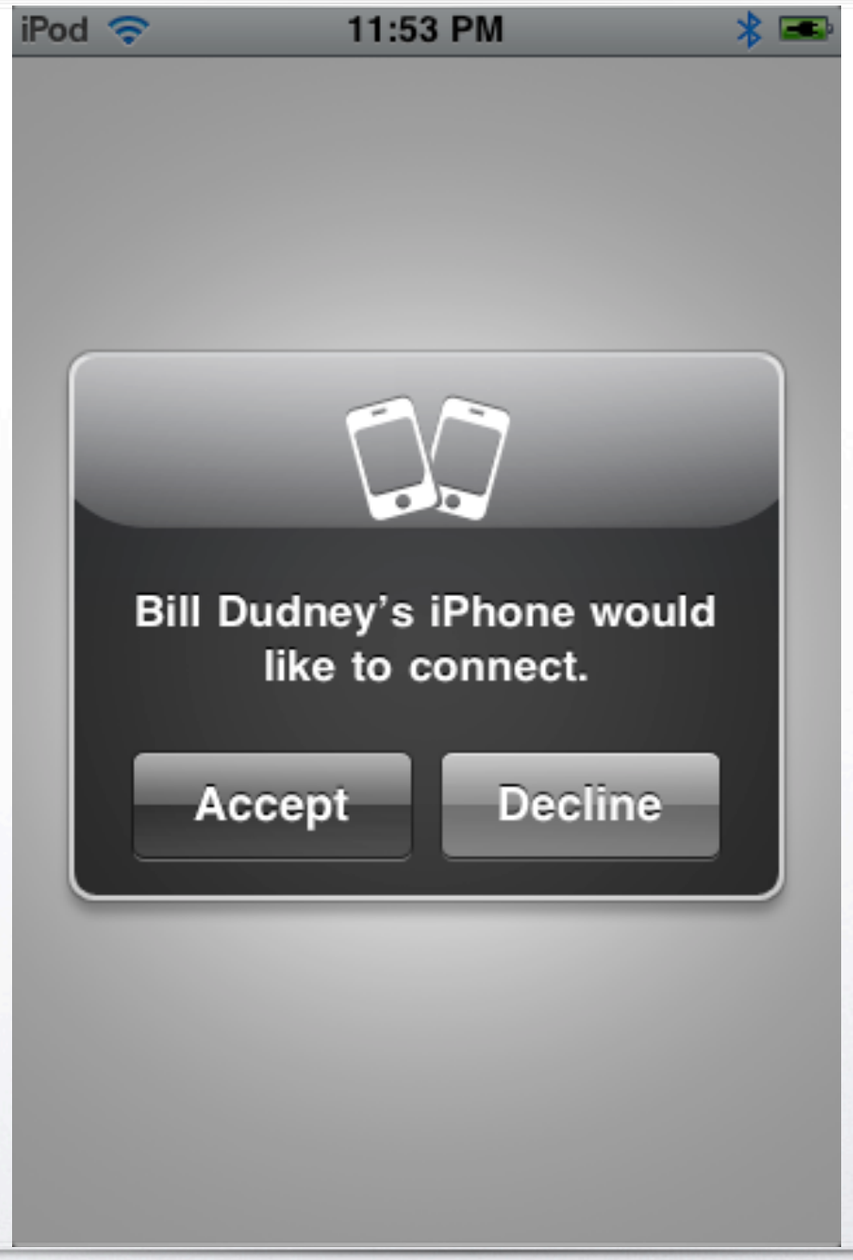
Application Startup

- Peer found and selected

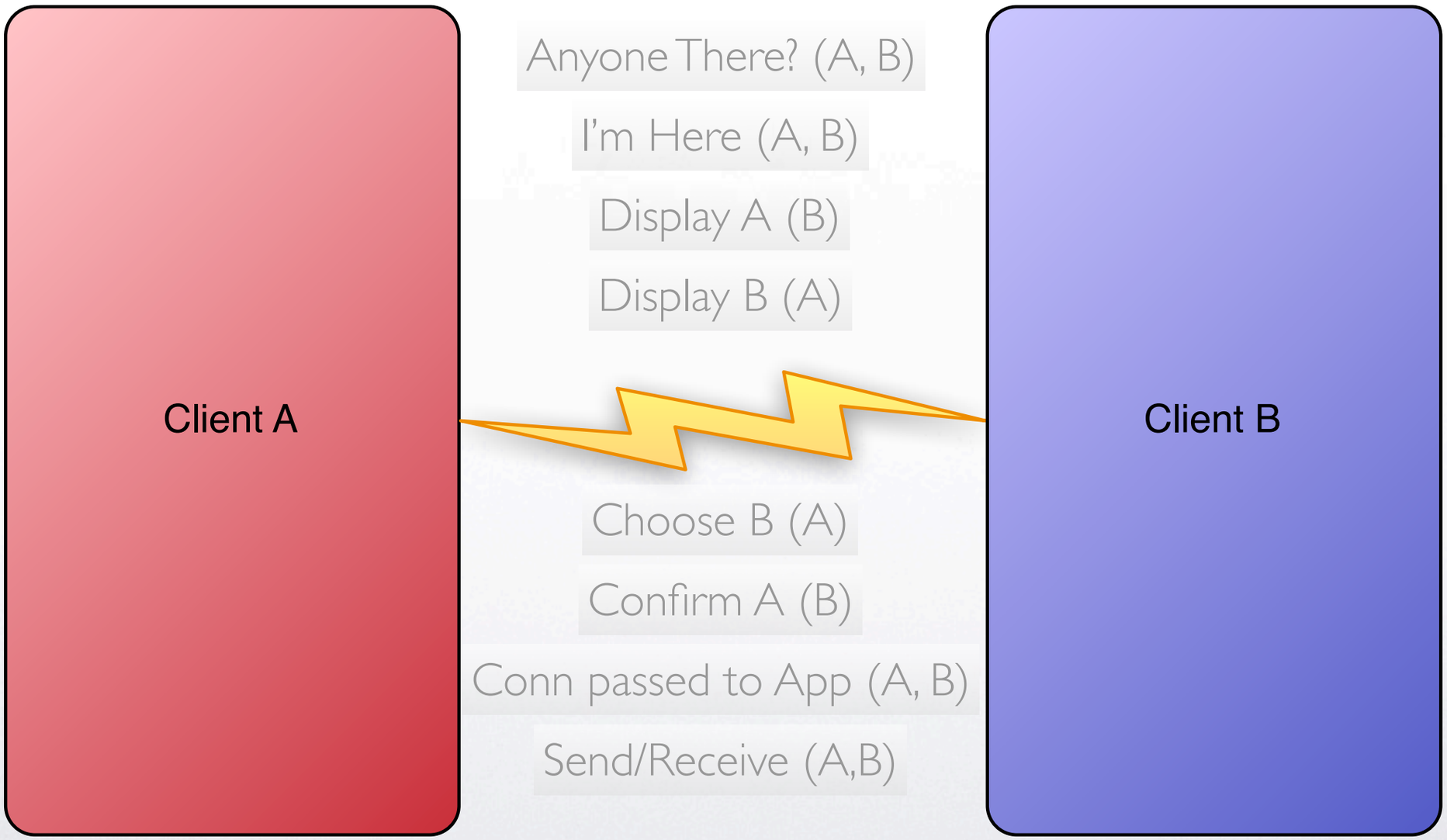


Application Startup

- Shall we play a game?

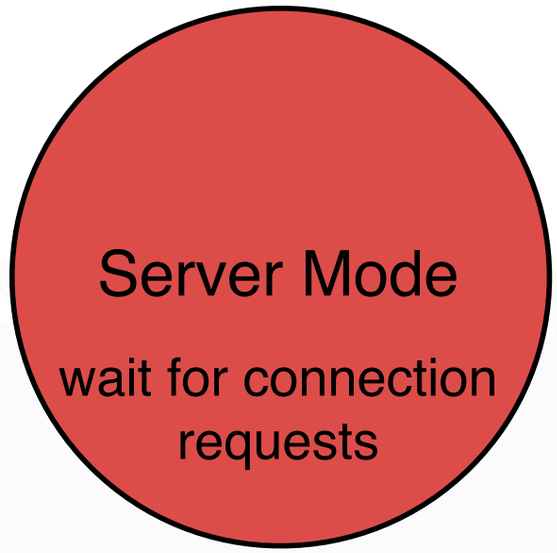


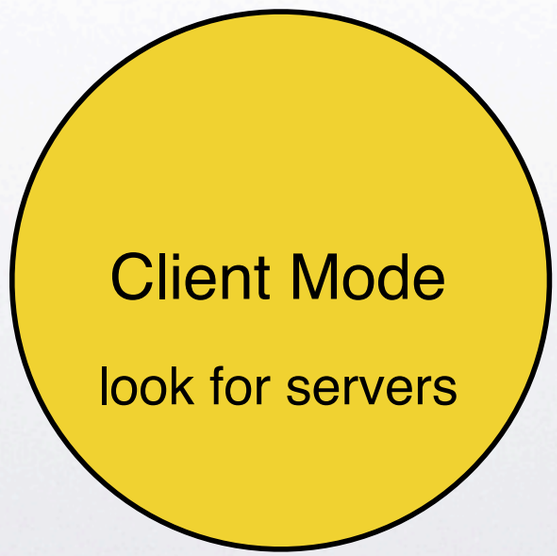
Peer Picker Process



Connect Peers

Connect





Connect Peers

Connect

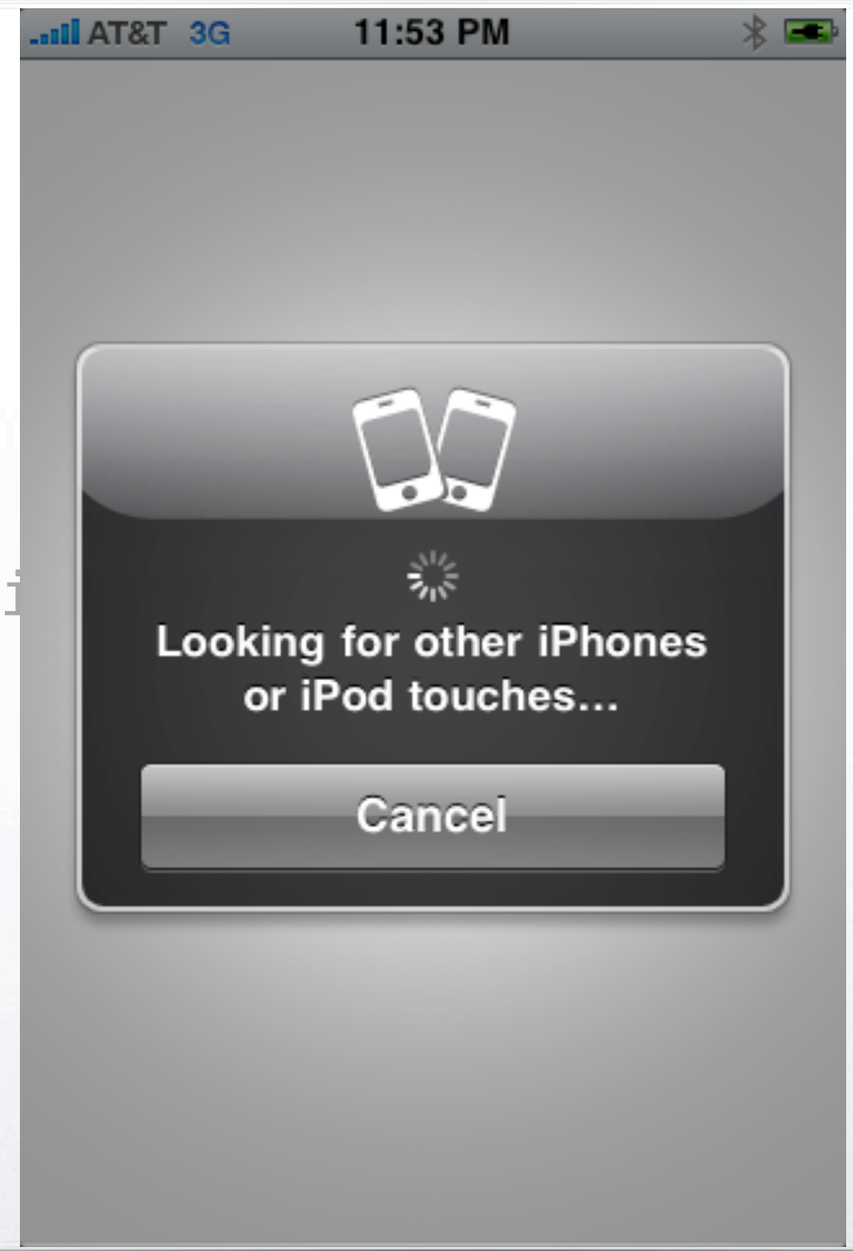
Connect Peers

Peer Picker Controller

```
- (void)startPeerPicker {
 GKPeerPickerController *picker =
        [[GKPeerPickerController alloc] init];
 picker.delegate = self;
 [picker show];
}
```

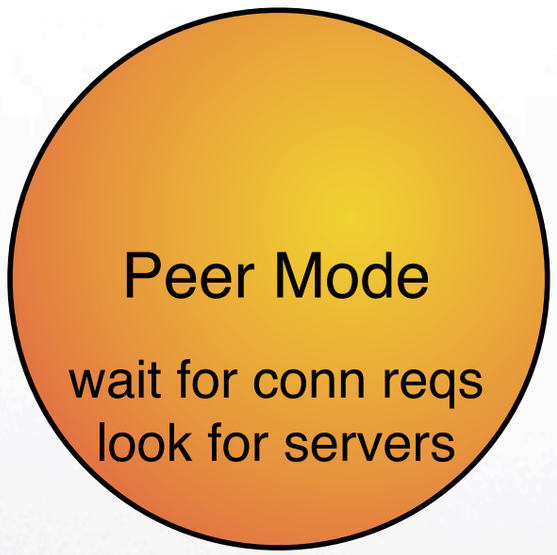
Peer Picker Controller

```
- (void)startPeerPicker {
 GKPeerPickerController *picker =
        [[GKPeerPickerController alloc] i
 picker.delegate = self;
 [picker show];
}
```



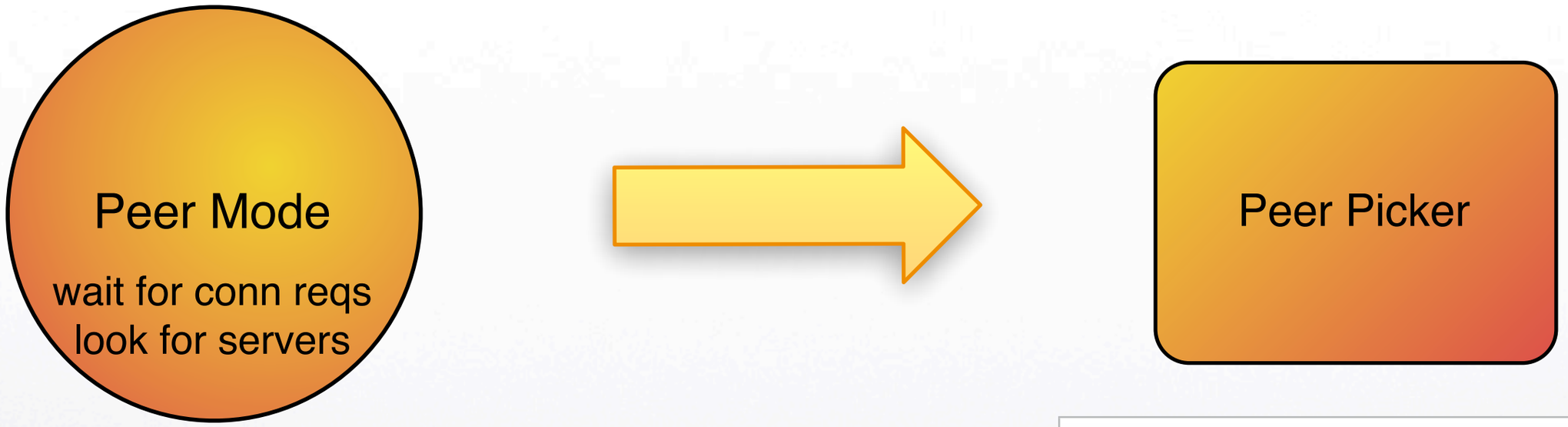
Connect Peers

Connect

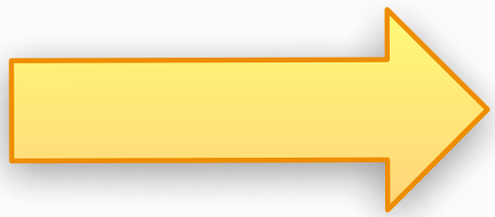


Connect Peers

Connect


Peer Mode
wait for conn reqs
look for servers
Peer Picker
Bill's iPod
Bill's 2G iPod
Bill's 1G iPod

Connect

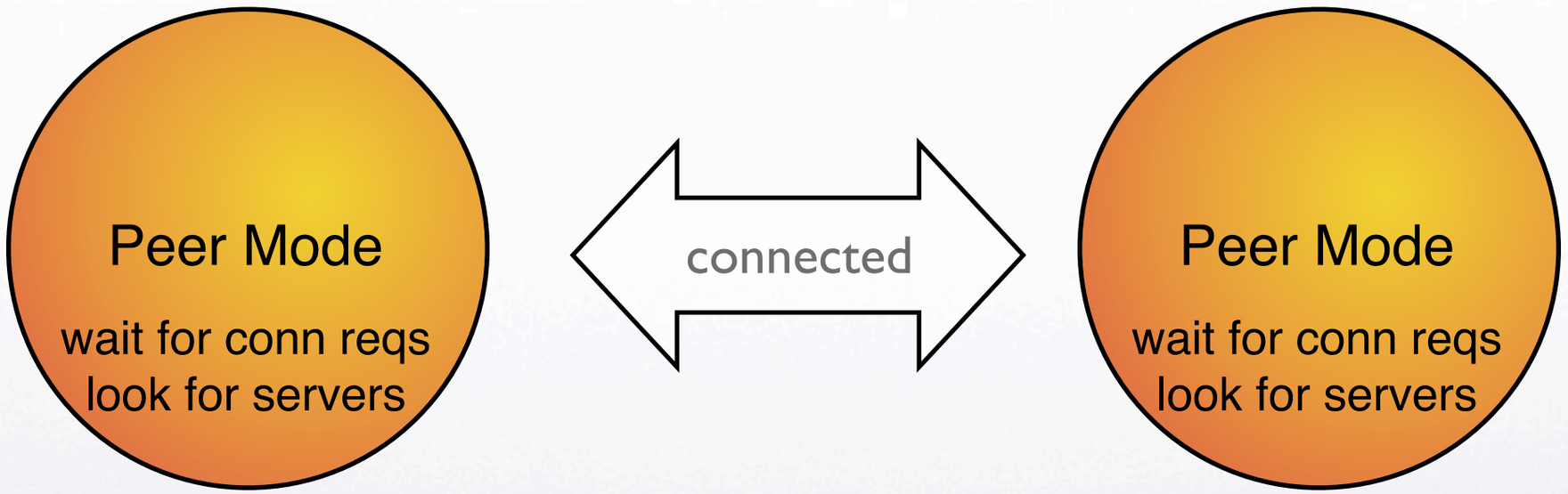


Tap

Bill's iPod
Bill's 2G iPod
Bill's 1G iPod

Connect Peers

Connect



Connect Peers

Session

```
- (GKSession *)peerPickerController:(GKPeerPickerController *)picker
            sessionForConnectionType:(GKPeerPickerConnectionType)type {
  return [[[GKSession alloc]
          initWithSessionID:@"_net.dudney.examples_"
                displayName:nil
                sessionMode:GKSessionModePeer] autorelease];
}
```

Make the Connection

```
- (void)peerPickerController:(GKPeerPickerController *)picker
              didConnectPeer:(NSString *)peerId
                   toSession:(GKSession *)session {
  if([peerId hash] > [session.peerID hash]) {
    _status = kServerGameStartedClientStatus;
  } else {
    _status = kServerGameStartedServerStatus;
  }
  self.peerId = peerId;
  self.gameSession = session;
  self.gameSession.delegate = self;
  [self.gameSession setDataReceiveHandler:self withContext:NULL];

  [picker dismiss];
  picker.delegate = nil;
  [picker autorelease];
}
```

Connect Peers

Make the Connection

```
- (void)peerPickerController:(GKPeerPickerController *)picker
              didConnectPeer:(NSString *)peerId
                   toSession:(GKSession *)session {
  if([peerId hash] > [session.peerID hash]) {
    _status = kServerGameStartedClientStatus;
  } else {
    _status = kServerGameStartedServerStatus;
  }
  self.peerId = peerId;
  self.gameSession = session;
  self.gameSession.delegate = self;
  [self.gameSession setDataReceiveHandler:self withContext:NULL];

  [picker dismiss];
  picker.delegate = nil;
  [picker autorelease];
}
```

Connect Peers

Loose Connection?

```
- (void)session:(GKSession *)session
           peer:(NSString *)peerID
 didChangeState:(GKPeerConnectionState)state {
  if(_status != kServerPickerShownStatus) {
    if(GKPeerStateDisconnected == state) {
      // got disconnected from the other peer
      NSString *message = [NSString stringWithFormat:@"Could not reconnect with %@.",
                           [session displayNameForPeer:peerID]];
      UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Lost Connection"
                                                      message:message
                                                     delegate:self
                                            cancelButtonTitle:@"End Game"
                                            otherButtonTitles:nil];
      [alert show];
      [alert release];
    }
  }
}
```


Session

```
- (void)alertView:(UIAlertView *)alertView
clickedButtonAtIndex:(NSInteger)buttonIndex {
  // index zero is the 'End Game' button, the only one
  if(buttonIndex == 0) {
    _status = kServerNotStartedStatus;
  }
}
```

Game Loop

```
- (id) init {
  self = [super init];
  if (self != nil) {
...
    self.link = [CADisplayLink displayLinkWithTarget:self
                                             selector:@selector(gameLoop)];
    self.link.frameInterval = 2;
...
  }
  return self;
}


- (void)start {
  [self.link addToRunLoop:[NSRunLoop mainRunLoop]
                  forMode:NSDefaultRunLoopMode];
}
```

Run Game Loop

Game Loop

- Start the peer picker
- Send new disk messages

Start the Picker

```
- (void)gameLoop {
  _loopId++;
  if(kServerNotStartedStatus == _status) {
    _status = kServerPickerShownStatus;
    [self startPeerPicker];
  } else if(kServerGameStartedServerStatus == _status) {
    if(0 == (_loopId % _diskFrequency)) {
      srandom(_loopId);
      long x = random() % 300 + 10;
      long y = random() % 460 + 10;
      CGPoint location = CGPointMake((CGFloat)x, (CGFloat)y);
      long colorIndex = random() % _colors.count;
      NSMutableData *data = [NSMutableData dataWithBytes:&location
                                                  length:sizeof(location)];
      [data appendBytes:&colorIndex length:sizeof(colorIndex)];
      [self sendMessageTypeID:kServerNewDiskMessageType data:data];
      UIColor *color = [_colors objectAtIndex:colorIndex];
      [self.delegate diskAtPoint:location withColor:color];
    }
  }
}
```

Run Game Loop

Send New Disks

```
- (void)gameLoop {
  _loopId++;
  if(kServerNotStartedStatus == _status) {
    _status = kServerPickerShownStatus;
    [self startPeerPicker];
  } else if(kServerGameStartedServerStatus == _status) {
    if(0 == (_loopId % _diskFrequency)) {
      srandom(_loopId);
      long x = random() % 300 + 10;
      long y = random() % 460 + 10;
      CGPoint location = CGPointMake((CGFloat)x, (CGFloat)y);
      long colorIndex = random() % _colors.count;
      NSMutableData *data = [NSMutableData dataWithBytes:&location
                                                  length:sizeof(location)];
      [data appendBytes:&colorIndex length:sizeof(colorIndex)];
      [self sendMessageTypeID:kServerNewDiskMessageType data:data];
      UIColor *color = [_colors objectAtIndex:colorIndex];
      [self.delegate diskAtPoint:location withColor:color];
    }
  }
}
```

Send New Disks

```
- (void)gameLoop {
  _loopId++;
  if(kServerNotStartedStatus == _status) {
    _status = kServerPickerShownStatus;
    [self startPeerPicker];
  } else if(kServerGameStartedServerStatus == _status) {
    if(0 == (_loopId % _diskFrequency)) {
      srandom(_loopId);
      long x = random() % 300 + 10;
      long y = random() % 460 + 10;
      CGPoint location = CGPointMake((CGFloat)x, (CGFloat)y);
      long colorIndex = random() % _colors.count;
      NSMutableData *data = [NSMutableData dataWithBytes:&location
                                                  length:sizeof(location)];
      [data appendBytes:&colorIndex length:sizeof(colorIndex)];
      [self sendMessageTypeID:kServerNewDiskMessageType data:data];
      UIColor *color = [_colors objectAtIndex:colorIndex];
      [self.delegate diskAtPoint:location withColor:color];
    }
  }
}
```

Send New Disks

```
- (void)gameLoop {
  _loopId++;
  if(kServerNotStartedStatus == _status) {
    _status = kServerPickerShownStatus;
    [self startPeerPicker];
  } else if(kServerGameStartedServerStatus == _status) {
    if(0 == (_loopId % _diskFrequency)) {
      srandom(_loopId);
      long x = random() % 300 + 10;
      long y = random() % 460 + 10;
      CGPoint location = CGPointMake((CGFloat)x, (CGFloat)y);
      long colorIndex = random() % _colors.count;
      NSMutableData *data = [NSMutableData dataWithBytes:&location
                                                  length:sizeof(location)];
      [data appendBytes:&colorIndex length:sizeof(colorIndex)];
      [self sendMessageTypeID:kServerNewDiskMessageType data:data];
      UIColor *color = [_colors objectAtIndex:colorIndex];
      [self.delegate diskAtPoint:location withColor:color];
    }
  }
}
```

Send New Disks

```
- (void)gameLoop {
  _loopId++;
  if(kServerNotStartedStatus == _status) {
    _status = kServerPickerShownStatus;
    [self startPeerPicker];
  } else if(kServerGameStartedServerStatus == _status) {
    if(0 == (_loopId % _diskFrequency)) {
      srandom(_loopId);
      long x = random() % 300 + 10;
      long y = random() % 460 + 10;
      CGPoint location = CGPointMake((CGFloat)x, (CGFloat)y);
      long colorIndex = random() % _colors.count;
      NSMutableData *data = [NSMutableData dataWithBytes:&location
                                                  length:sizeof(location)];
      [data appendBytes:&colorIndex length:sizeof(colorIndex)];
      [self sendMessageTypeID:kServerNewDiskMessageType data:data];
      UIColor *color = [_colors objectAtIndex:colorIndex];
      [self.delegate diskAtPoint:location withColor:color];
    }
  }
}
```

Send New Disks

```
- (void)gameLoop {
  _loopId++;
  if(kServerNotStartedStatus == _status) {
    _status = kServerPickerShownStatus;
    [self startPeerPicker];
  } else if(kServerGameStartedServerStatus == _status) {
    if(0 == (_loopId % _diskFrequency)) {
      srandom(_loopId);
      long x = random() % 300 + 10;
      long y = random() % 460 + 10;
      CGPoint location = CGPointMake((CGFloat)x, (CGFloat)y);
      long colorIndex = random() % _colors.count;
      NSMutableData *data = [NSMutableData dataWithBytes:&location
                                                  length:sizeof(location)];
      [data appendBytes:&colorIndex length:sizeof(colorIndex)];
      [self sendMessageTypeID:kServerNewDiskMessageType data:data];
      UIColor *color = [_colors objectAtIndex:colorIndex];
      [self.delegate diskAtPoint:location withColor:color];
    }
  }
}
```

Run Game Loop

Send New Disks

```
- (void)gameLoop {
  _loopId++;
  if(kServerNotStartedStatus == _status) {
    _status = kServerPickerShownStatus;
    [self startPeerPicker];
  } else if(kServerGameStartedServerStatus == _status) {
    if(0 == (_loopId % _diskFrequency)) {
      srandom(_loopId);
      long x = random() % 300 + 10;
      long y = random() % 460 + 10;
      CGPoint location = CGPointMake((CGFloat)x, (CGFloat)y);
      long colorIndex = random() % _colors.count;
      NSMutableData *data = [NSMutableData dataWithBytes:&location
                                                  length:sizeof(location)];
      [data appendBytes:&colorIndex length:sizeof(colorIndex)];
      [self sendMessageTypeID:kServerNewDiskMessageType data:data];
      UIColor *color = [_colors objectAtIndex:colorIndex];
      [self.delegate diskAtPoint:location withColor:color];
    }
  }
}
```

Send New Disks

```
- (void)gameLoop {
  _loopId++;
  if(kServerNotStartedStatus == _status) {
    _status = kServerPickerShownStatus;
    [self startPeerPicker];
  } else if(kServerGameStartedServerStatus == _status) {
    if(0 == (_loopId % _diskFrequency)) {
      srandom(_loopId);
      long x = random() % 300 + 10;
      long y = random() % 460 + 10;
      CGPoint location = CGPointMake((CGFloat)x, (CGFloat)y);
      long colorIndex = random() % _colors.count;
      NSMutableData *data = [NSMutableData dataWithBytes:&location
                                                  length:sizeof(location)];
      [data appendBytes:&colorIndex length:sizeof(colorIndex)];
      [self sendMessageTypeID:kServerNewDiskMessageType data:data];
      UIColor *color = [_colors objectAtIndex:colorIndex];
      [self.delegate diskAtPoint:location withColor:color];
    }
  }
}
```

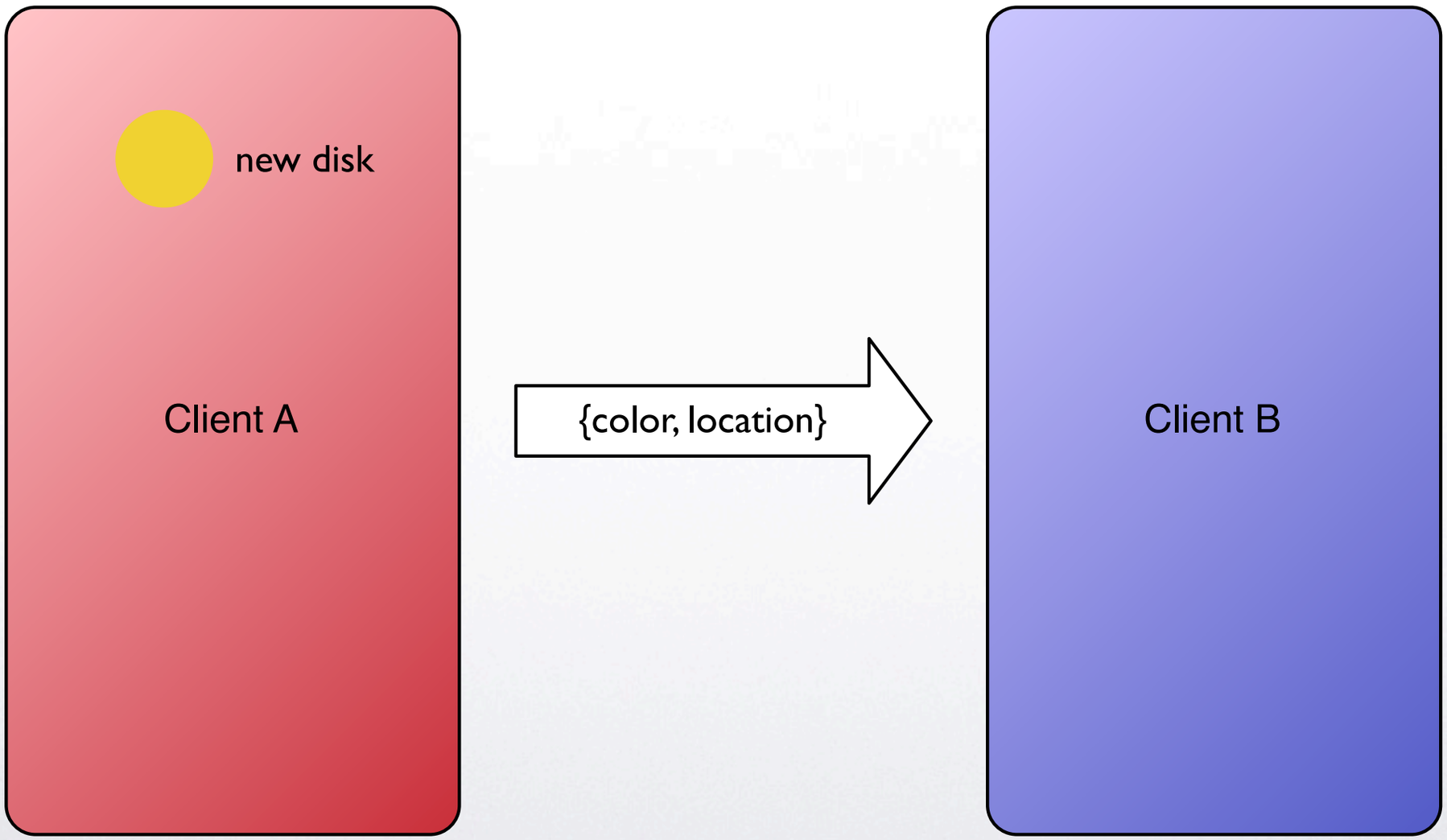
Send Data

```
- (void)sendMessageTypeID:(ServerPacketTypeId)packetTypeId data:(NSData *)data {
  NSMutableData *sentData = [NSMutableData data];
  [sentData appendBytes:&packetTypeId length:sizeof(packetTypeId)];
  [sentData appendData:data];
  [self.gameSession sendData:sentData
                      toPeers:[NSArray arrayWithObject:self.peerId]
                 withDataMode:GKSendDataUnreliable error:nil];
}
```

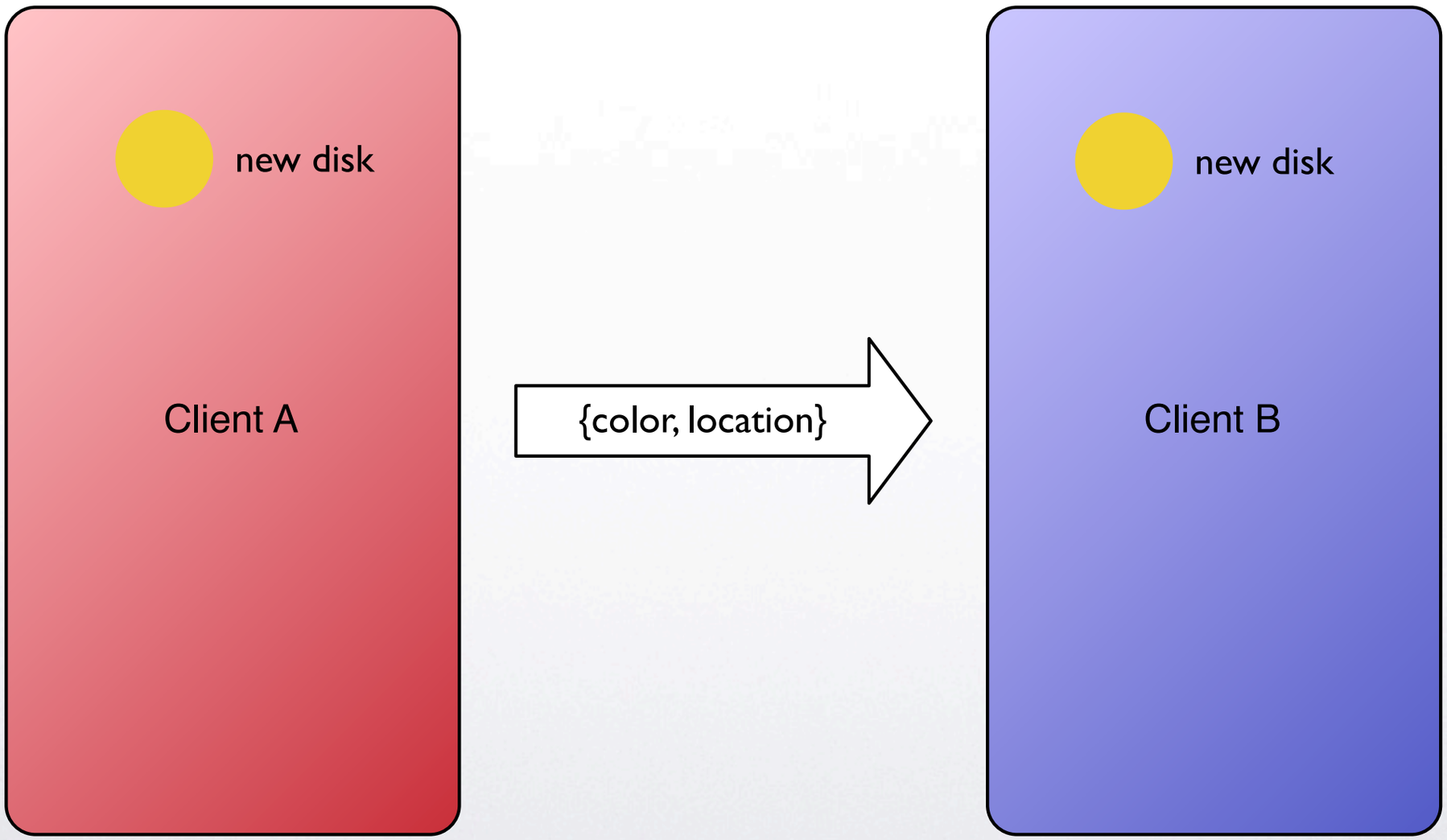
Receive Data

```
- (void) receiveData:(NSData *)data
            fromPeer:(NSString *)peer
           inSession:(GKSession *)session
             context:(void *)context {
  ServerPacketTypeId typeId;
  [data getBytes:&typeId length:sizeof(typeId)];
  if(kServerNewDiskMessageType == typeId) {
    CGPoint point;
    NSRange range;
    range.location = sizeof(typeId);
    range.length = sizeof(point);
    [data getBytes:&point range:range];
    NSUInteger colorIndex = 0;
    range.location = range.location + range.length;
    range.length = sizeof(colorIndex);
    [data getBytes:&colorIndex range:range];
    [self.delegate diskAtPoint:point
                     withColor:[_colors objectAtIndex:colorIndex]];
  } else if(kServerSmashMessageType == typeId) {
...
  }
}
```

Messages



Messages



New Disk Layer

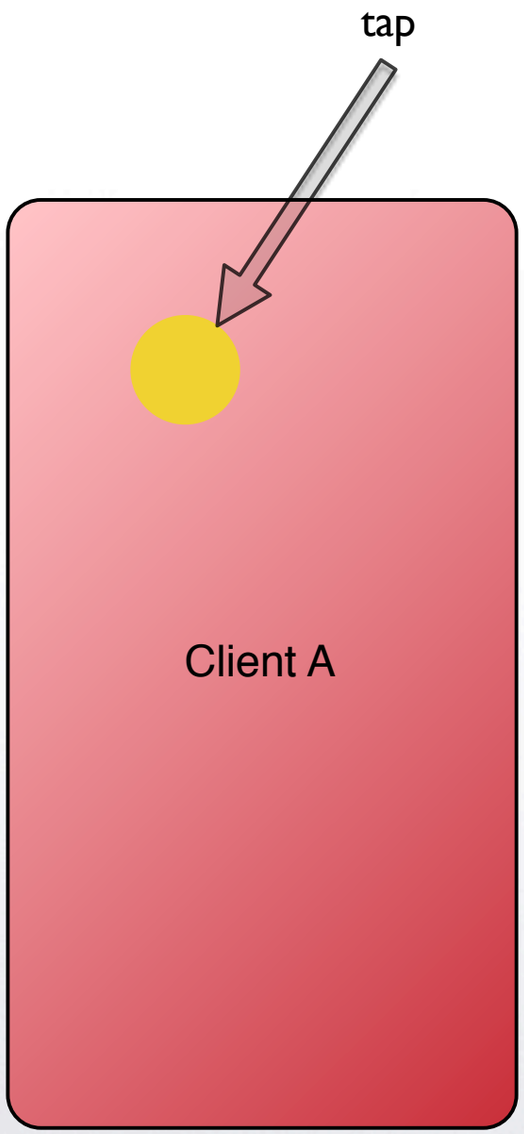
```
- (void)displayDiskAt:(CGPoint)point ofColor:(UIColor *)color {
   DiskLayer *layer = [DiskLayer layerWithColor:color];
   layer.position = point;
   [self.layer addSublayer:layer];
   [self performSelector:@selector(disapearLayer:)
                withObject:layer afterDelay:0.1f];
 }

- (void)disapearLayer:(CALayer *)layer {
   layer.opacity = 0.0f;
}
```

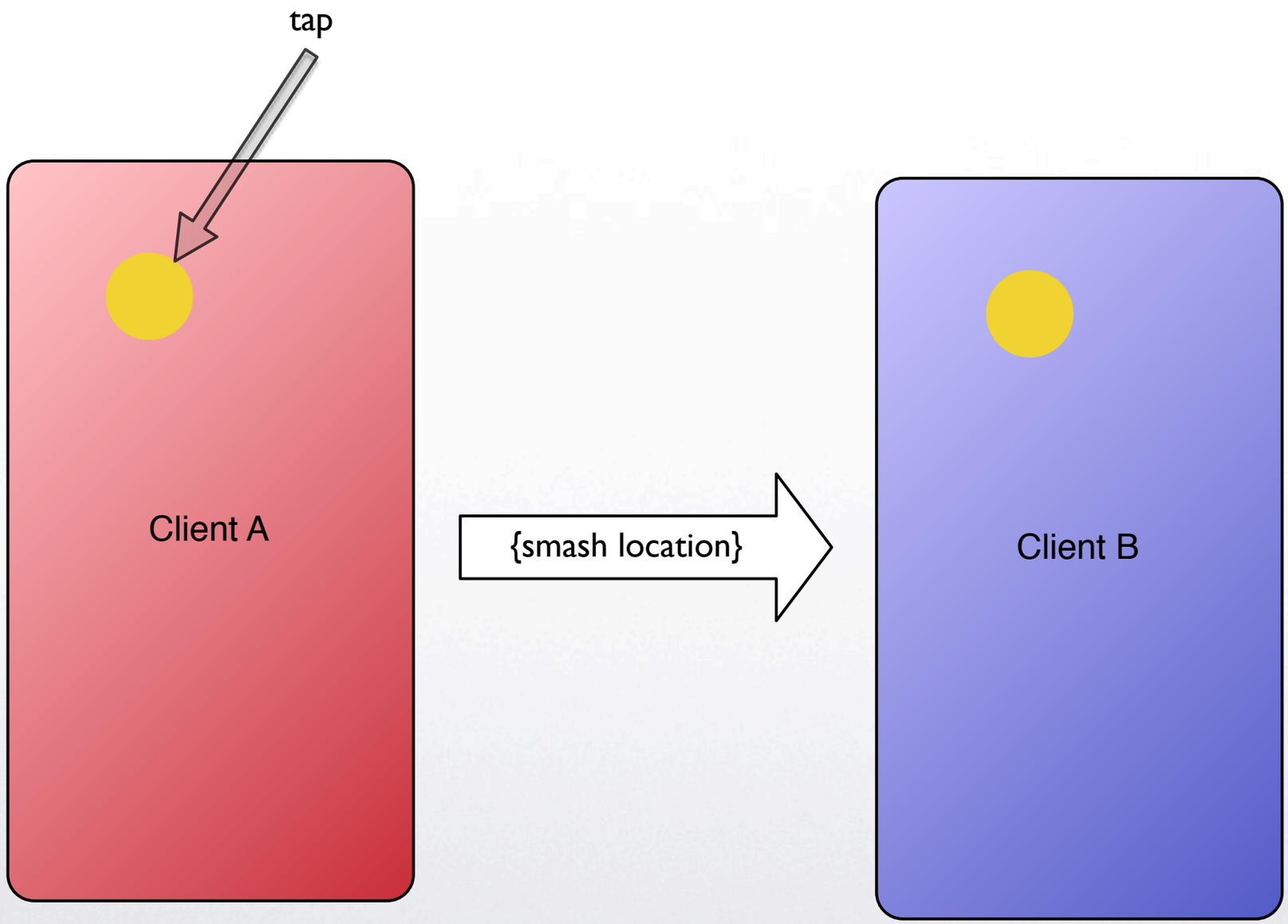
New Disk Layer

```
- (id) initWithColor:(UIColor *)color {
  self = [super init];
  if (self != nil) {
    CGRect frame = CGRectMake(0.0f, 0.0f, 40.0f, 40.0f);
    self.frame = frame;
    self.diskLayer = [CAShapeLayer layer];
    self.diskLayer.frame = frame;
    CGPathRef path = [self newCirclePath];
    self.diskLayer.path = path;
    self.diskLayer.fillColor = [color CGColor];
    self.diskLayer.strokeColor = [color CGColor];
    CFRelease(path);
    CABasicAnimation *animation = [CABasicAnimation animationWithKeyPath:@"opacity"];
    animation.duration = 4.9f;
    animation.delegate = self;
    self.actions = [NSDictionary dictionaryWithObject:animation forKey:@"opacity"];
    [self addSublayer:self.diskLayer];
    _smashed = NO;
  }
  return self;
}
```

Smash A Disk



Smash A Disk



Process Events

```
- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event {
   CGPoint point = [[touches anyObject] locationInView:self];
   CGPoint location = [self.layer convertPoint:point
                                       toLayer:self.layer.superlayer];
   CALayer *layer = [self.layer hitTest:location];
   if(layer != self.layer) {
      [self.delegate diskSmashedAt:location];
      [(DiskLayer *)layer smash];
   }
}
```

Smashing Disks

```
- (void)diskSmashedAt:(CGPoint)point {
    [self.server sendSmashAtPoint:point];
}
```

Sending Smashes

```
- (void)sendSmashAtPoint:(CGPoint)point {
  NSData *data = [NSData dataWithBytes:&point length:sizeof(point)];
  [self sendMessageTypeID:kServerSmashMessageType data:data];
}
```

Receive Smash

```
- (void) receiveData:(NSData *)data
            fromPeer:(NSString *)peer
           inSession:(GKSession *)session
             context:(void *)context {

...
  } else if(kServerSmashMessageType == typeId) {
    CGPoint point;
    NSRange range;
    range.location = sizeof(typeId);
    range.length = sizeof(point);
    [data getBytes:&point range:range];
    [self.delegate remoteDiskSmashedAtPoint:point];
  }
}
```

Remote Smash Forward

```
- (void)remoteDiskSmashedAtPoint:(CGPoint)point {
    DiskSmashView *smashView = (DiskSmashView *)self.view;
    [smashView remoteSmash:point];
}
```

Remote Smash

```
- (void)remoteSmash:(CGPoint)point {
   CALayer *layer = [self.layer hitTest:point];
   if(layer != self.layer) {
      [(DiskLayer *)layer smash];
   }
}
```

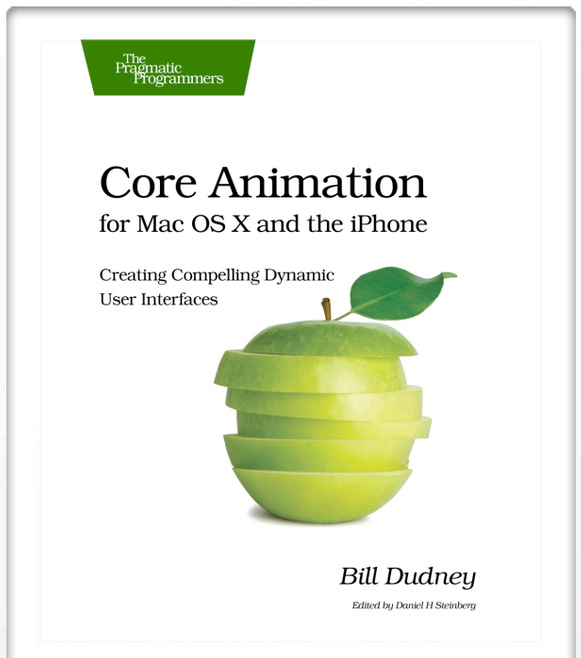
Conclusion

- Getting Connected is easy!
- Sending Data is easy
- Designing a good game, not so much.

Thanks!

Pragmatic iPhone Studio

Get hands-on
Cocoa & iPhone Training
in The Pragmatic Studio



PragProg.com discount code
voices_bill_dudney_10