```c
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include "List.h"
#include "Graph.h"

/*** Private Function Prototypes ***/
int insertEdge( ListRef L, int u );
int isInOrderRange( int u, int order );
void killGraph( char* e );

/*** Graph Constructor / Destructor ***/
GraphRef newGraph( int n ){
   int i;
   GraphRef G = malloc( sizeof(Graph) );
   assert( G != NULL );
   G->order = n;
   G->size = G->source = NIL;

   G->adj = malloc( (n+1) * sizeof(ListRef*) );
   assert( G->adj != NULL );

   G->color = calloc( (n+1),  sizeof(int) );
   assert( G->color != NULL );

   G->d = calloc( (n+1) , sizeof(int) );
   assert( G->d != NULL );

   G->P = calloc( n+1, sizeof(int) );
   assert( G->P != NULL );

   for ( i = 1; i <= n; i++ ){
      G->adj[i] = newList();
      G->d[i] = INF;
   }
   return G;
}

void freeGraph( GraphRef* pG ){
   int i;
   if (pG != NULL && *pG != NULL ){
      for ( i = 1; i <= getOrder((*pG)); i++ ){
         freeList( &((*pG)->adj[i]) );
      }
      free((*pG)->adj);
```



```c
      free((*pG)->color);
      free((*pG)->d);
      free((*pG)->P);
      free(*pG);
      *pG = NULL;
   }
}

/*** Access Functions ***/
int getOrder( GraphRef G ) {
   if ( G == NULL ) killGraph("Calling getOrder() on NULL GraphRef");
   return G->order;
}

int getSize( GraphRef G ) {
   if ( G == NULL ) killGraph("Calling getSize() on NULL GraphRef");
   return G->size;
}

int getSource( GraphRef G ) {
   if ( G == NULL ) killGraph("Calling getSource() on NULL GraphRef");
   if ( G->source == NIL ) return NIL;
   else return G->source;
}

int getParent( GraphRef G, int u ){
   if ( G == NULL ) killGraph("Calling getParent() on NULL GraphRef");
   if ( !isInOrderRange(u, getOrder(G) ) )
      killGraph("Method getParent() requires an input vertex u \
such that 1 <= u <= Order of graph");

   if ( G->source == NIL ) return NIL;
   else return G->P[u];
}

int getDist( GraphRef G, int u ){
   if ( G == NULL ) killGraph("Calling getDist() on NULL GraphRef");
   if ( !isInOrderRange(u, getOrder(G) ) )
      killGraph("Method getDist() requires an input vertex u \
such that 1 <= u <= Order of graph");
   if ( G->source == NIL ) return INF;
   else return G->d[u];
}

void getPath( ListRef L, GraphRef G, int u ) {
```

```
      if ( G == NULL ) killGraph("Calling getPath() on NULL GraphRef");
      if ( !isInOrderRange(u, getOrder(G) ) )
         killGraph("Method getPath() requires an input vertex u \
   such that 1 <= u <= Order of graph");
      /*append NIL to list if no path exists*/
      if ( G->source == u ){
         insertFront(L,u);
      } else if ( G->P[u] == NIL ){
         insertFront(L,NIL); /*path doesn't exist*/
      } else {
         getPath(L,G,G->P[u]);
         insertBack(L,u);
      }
   }



   /*** Manipulation Procedures ***/
   void makeNull( GraphRef G ){
       if ( G == NULL ) killGraph("Calling addEdge() on NULL GraphRef");
      int i;
      for ( i = 1; i <= getOrder(G); i++ ){
         makeEmpty(G->adj[i]);
         G->d[i] = INF;
         G->P[i] = G->color[i] = NIL;
      }
      G->size = G->source = 0;
   }

   void addEdge( GraphRef G, int u, int v ){
      if ( G == NULL ) killGraph("Calling addEdge() on NULL GraphRef");
      if ( !isInOrderRange(u, getOrder(G) ) ||
               !isInOrderRange(v, getOrder(G) ) )
         killGraph("Method addEdge() requires precondition u and v within\
   range of 1 to the order of G");

      ListRef uList = G->adj[u];
      ListRef vList = G->adj[v];

      if ( insertEdge( vList, u ) && insertEdge( uList, v) )
         G->size += 1;

   }

   void addArc( GraphRef G, int u, int v ){
```



```
      if ( G == NULL ) killGraph("Calling addArc() on NULL GraphRef");
      if ( !isInOrderRange(u, getOrder(G) ) ||
               !isInOrderRange(v, getOrder(G) ) )
         killGraph("Method addArc() requires precondition u and v within\
   range of 1 to the order of G");

      if ( insertEdge(G->adj[u], v) ) G->size += 1;
   }

   void BFS( GraphRef G, int s ){
      int i, x, y;
      ListRef Q, L;
      G->source = s;
      for ( i = 1; i <= getOrder(G); i++ ){
         if ( i != s) {
            G->color[i] = WHITE;
            G->d[i] = INF;
            G->P[i] = NIL;
         }
      }
      G->color[s] = GREY;
      G->d[s] = G->P[s] = NIL;
      /* Q = FIFO queue, where enqueue = insertback, dequeue = delete front */
      Q = newList();
      insertBack(Q,s);
      while ( !isEmpty(Q) ){
         x = getFront(Q);
         deleteFront(Q);
         L = G->adj[x];
         moveTo(L,0);
         while( !offEnd(L) ){
            y = getCurrent(L);
            if ( G->color[y] == WHITE ){
               G->color[y] = GREY;
               G->d[y] = G->d[x]+1;
               G->P[y] = x;
               insertBack(Q,y);
            }
            moveNext(L);
         }
         G->color[x] = BLACK;
      }
      freeList(&Q);
   }
```

```
/* insertEdge() - adds u to the adjacency list L in sorted order,
   returns 0 if edge already exists in adj list, 1 otherwise.
*/
int insertEdge( ListRef L, int u ) {
   if ( isEmpty(L) ) {
      insertFront(L, u);
   }else {
      moveTo(L,0);
      int entry = getCurrent(L);
      while ( entry < u ){
         moveNext(L);
         if ( offEnd(L) ) { entry = NIL; break; }
         else entry = getCurrent(L);
      }
      if ( entry == NIL ) {
         insertBack(L,u);
      } else if ( entry > u ){
         insertBeforeCurrent(L,u);
      }else return 0;
      /* the only other case is that this edge already exists, do nothing */
   }
   return 1;
}

/*** Other functions ***/
void printGraph( FILE* out, GraphRef G ){
   if ( G == NULL ) killGraph("Calling printGraph() on NULL GraphRef");
   int i;
   for ( i = 1; i <= getOrder(G); i++ ){
      if ( !isEmpty( G->adj[i] ) ){
         fprintf( out, "%d:", i);
         moveTo(G->adj[i],0);
         while( !offEnd(G->adj[i]) ){
            fprintf(out, " %d", getCurrent( G->adj[i]) );
            moveNext( G->adj[i] );
         }
         fprintf( out, "\n" );
      }
   }
}

/* killGraph() - prints error e to stdout and exits program */
void killGraph( char* e ) {
   printf( "Graph.c: %s\n", e);
   exit(1);
```



```
}

void printGraphInfo( GraphRef G ){
   if ( G == NULL ) killGraph("Calling printGraphInfo() on NULL GraphRef");
   printf("Graph G has\nsize %d\norder %d \
           \nsource %d\n",getSize(G),getOrder(G),getSource(G));
}

/* isInOrderRange() - returns true if 1 <= u <= order. false otherwise. */
int isInOrderRange( int u, int order ){
   if ( u < 1 || u > order) return 0;
   else return 1;
}
```

```
#ifndef __GRAPHC_H__
#define __GRAPHC_H__

#define INF    -1
#define NIL    0
#define WHITE  1
#define GREY   2
#define BLACK  3

#include "List.h"

typedef struct Graph{
   ListRef* adj;/*array of lists who's ith element contains
                  neighbors of vertex i */
   int* color; /* color of x = color[x]
                  such that white = 1, grey = 2, black = 3 */
   int* d; /* distance from source to x = d[x] */
   int* P; /* parent of x = P[x] */
   int order; /* # of vertices */
   int size; /* # of edges */
   int source; /* last vertex used by BFS */
}Graph;

typedef struct Graph* GraphRef;

/*** Contructos / Destructors ***/
GraphRef newGraph( int n );
void freeGraph( GraphRef* pG );

/*** Access functions ***/
int getOrder( GraphRef G );
int getSize( GraphRef G );
int getSource( GraphRef G );
int getParent( GraphRef G, int u );
int getDist( GraphRef G, int u );
void getPath( ListRef L, GraphRef G, int u );

/*** Manipulation procedures ***/
void makeNull( GraphRef G );
void addEdge( GraphRef G, int u, int v );
void addArc( GraphRef G, int u, int v );
void BFS( GraphRef G, int s );


/*** Other operations ***/
```



```
void printGraph( FILE* out, GraphRef G );
void printGraphInfo( GraphRef G );

#endif
```

```c
/* $Id: List.c,v 1.4 2011-10-08 19:09:41-07 - - $ */

/*
 * List.c
 * A doubly linked list ADT for integers.
 * PA2
 * By B Stewart Bracken
 * bbracken@ucsc.edu
 * ID# 1187817
*/

#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include "List.h"

/* Private inner Node struct, corresponding reference type, and
      * * constructor-destructor pair. Not exported.  */
typedef struct Node{
   int data;
   struct Node* next;
   struct Node* prev;
} Node;

typedef Node* NodeRef;

/*Node constructor*/
NodeRef newNode(int node_data) {
   NodeRef N = malloc( sizeof(Node) );
   assert ( N != NULL );
   N->data = node_data;
   N->next = NULL;
   return (N);
}

/*Node deconstructor*/
void freeNode(NodeRef* pN) {
   if (pN != NULL && *pN != NULL){
      free(*pN);
      *pN = NULL;
   }
}


/* Public List struct, constructor-destructor */
```



```c
typedef struct List{
   NodeRef front, back, current;
   int length, index;
} List;

/* List constructor */
ListRef newList(void){
   ListRef L;
   L = malloc(sizeof(List));
   assert ( L!= NULL );
   L->front = L->back = L->current = NULL;
   L->length = 0;
   L->index = -1;
   return(L);
}

/* List deconstructor */
void freeList(ListRef* pL) {
   if ( pL != NULL && *pL != NULL ){
      if (!isEmpty(*pL) ) {
         /*free all the things!*/
         makeEmpty(*pL);
      }
      free(*pL);
      *pL = NULL;
   }
}


/*Access functions ***********************************************
*****************************************************************/

/* getLength() - Returns length of list. */
int getLength(ListRef L) {
   if ( L == NULL ) killProgram("Calling getLength() on NULL ListRef.");
   return (L->length);
}

/* isEmpty() - Returns true if this List is empty, false otherwise. */
int isEmpty(ListRef L) {
   if ( L == NULL ) killProgram("Calling isEmpty() on NULL ListRef.");
   return ( L->length == 0 );
}

/* offEnd() - Returns true if current is undefined. */
```

```
int offEnd(ListRef L) {
   if ( L == NULL ) killProgram("Calling offEnd() on NULL ListRef.");
   return (L->current == NULL);
}

/* getIndex() - Returns the current index position from 0 to
   length-1, or -1 if current is undefined. */
int getIndex(ListRef L) {
   if ( L == NULL ) killProgram("Calling getIndex() on NULL ListRef.");
   return (L->index);
}

/* getFront() - Returns front element.
   Pre: !isEmpty() */
int getFront(ListRef L) {
   if ( L == NULL ) killProgram("Calling getFront() on NULL ListRef.");
   if (isEmpty(L)) {
      killProgram("Method getFront() failed to pass pre !isEmpty() check.");
   } else {
      return (L->front->data);
   }
   return -1111;
}

/* getBack() - Returns back element.
   Pre: !isEmpty() */
int getBack(ListRef L) {
   if ( L == NULL ) killProgram("Calling getBack() on NULL ListRef.");
   if (isEmpty(L)) {
      killProgram("Method getBack() failed to pass pre !isEmpty() check.");
   } else {
      return (L->back->data);
   }
   return -1111;
}

/* getCurrent() - Returns current element.
   Pre: !isEmpty(), !offEnd() */
int getCurrent(ListRef L) {
   if ( L == NULL ) killProgram("Calling getCurrent() on NULL ListRef.");
    if (isEmpty(L) || offEnd(L)) {
        killProgram("Method getCurrent() failed to pass pre !isEmpty() && !offEnd()
check.");
    } else {
        return L->current->data;
```



```
    }
   return -1111;
}

/* equals() - Returns true if this List and L are the same integer
   sequence. Ignores the current element in both Lists. */
int equals(ListRef L, ListRef M) {
   if ( L == NULL || M == NULL ) killProgram("Calling equals() on NULL ListRef.");
   NodeRef currL = L->front;
   NodeRef currM = M->front;
   if ( getLength(L) != getLength(M) ) return FALSE;
   while ( currL != NULL && currM != NULL ) {
      if ( currL->data != currM->data ) return FALSE;
      currL = currL->next;
      currM = currM->next;
   }
   return TRUE;
}

/* Manipulation procedures ***************************************
*****************************************************************/

/* makeEmpty() - Sets this List to the empty state.
   Post: isEmpty(). */
void makeEmpty(ListRef L){
   if ( L == NULL ) killProgram("Calling makeEmpty() on NULL ListRef.");
   if (!isEmpty(L) ) {
      /*free all the things!*/
      while(!isEmpty(L)){
         deleteFront(L);
      }
      L->index = -1;
      L->front = L->back = L->current = NULL;
      L->length = 0;
   }
}

/* moveTo() - Moves current element marker to position i in
   this List. */
void moveTo(ListRef L, int i){
   if ( L == NULL ) killProgram("Calling moveTo() on NULL ListRef.");
   if ( i < 0 || i >= getLength(L) )
      killProgram("Bad index pass in moveTo() method.");
   else {
      NodeRef N;
```

```
        int k;
        int distFromCurrent = abs(L->index-i);
        if ( i == 0 ) {
           N = L->front;
        }else if ( i == getLength(L)-1 ) {
           N = L->back;
        } else if (i == L->index) {
           N = L->current;
        }else if ( (L->index > 0) && (distFromCurrent < i)
                    && (distFromCurrent < (getLength(L)-1-i)) ) {
           if ( L->index - i > 0 ) {
              for ( N = L->current, k = L->index; k > i; k--, N = N->prev);
           } else {
              for ( N = L->current, k = L->index; k < i; k++, N = N->next);
           }
        } else if ( (getLength(L)-1-i) <= i) {
           for ( N = L->back, k = getLength(L)-1; k>i; k--, N = N->prev);
        } else {
           for ( N = L->front, k = 0; k < i; k++, N = N->next);
        }
        L->current = N;
        L->index = i;
     }
  }

  /* movePrev() - Moves current one step toward front element.
     Pre: !isEmpty(), !offEnd(). */
  void movePrev(ListRef L) {
     if ( L == NULL ) killProgram("Calling movePrev() on NULL ListRef.");
      if (isEmpty(L) || offEnd(L)) {
          killProgram("Method movePrev() failed to pass pre !isEmpty() && !offEnd()
  check.");
      } else {
          L->current = L->current->prev;
          L->index--;
      }
  }

  /* moveNext() - Moves current one step toward back element.
     Pre: !isEmpty(), !offEnd(). */
  void moveNext(ListRef L) {
     if ( L == NULL ) killProgram("Calling moveNext() on NULL ListRef.");
      if (isEmpty(L) || offEnd(L)) {
          killProgram("Method moveNext() failed to pass pre !isEmpty() && !offEnd()
  check.");
```



```
      } else {
          L->current = L->current->next;
          L->index++;
      }
  }

  /* insertFront() - Inserts new element at the front position.
     Post: !isEmpty(). */
  void insertFront(ListRef L, int data) {
     if ( L == NULL ) killProgram("Calling insertFront() on NULL ListRef.");
      NodeRef N = newNode(data);
      if (L->front == NULL) {
          L->front = N;
          L->back = N;
      } else {
          L->front->prev = N;
          N->next = L->front;
      }
      L->front = N;
      L->length++;

      if (isEmpty(L)) {
          killProgram("Method insertFront() failed to pass post isEmpty() check.");
      }
  }

  /* insertBack() - Inserts new element in the back position.
     Post: !isEmpty(). */
  void insertBack(ListRef L, int data) {
     if ( L == NULL ) killProgram("Calling insertBack() on NULL ListRef.");
      NodeRef N = newNode(data);
      if (L->back == NULL) {
          L->front = N;
          L->back = N;
      } else {
          L->back->next = N;
          N->prev = L->back;
      }
      L->back = N;
      L->length++;

      if (isEmpty(L)) {
          killProgram("Method insertBack() failed to pass post isEmpty() check.");
      }
  }
```

```
/* insertBeforeCurrent() - Inserts new element before current element.
    increments index by 1.
    Pre: !isEmpty(), !offEnd() */
void insertBeforeCurrent(ListRef L, int data) {
   if ( L == NULL ) killProgram("Calling insertBeforeCurrent() on NULL ListRef.");
    NodeRef N = newNode(data);
    if (isEmpty(L)) {
        killProgram("Method insertBeforeCurrent() failed to pass pre isEmpty()
check.");
    } else if (offEnd(L) ){
        killProgram("Method insertBeforeCurrent() failed to pass pre offEnd()
check.");
    } else{
        N->prev = L->current->prev;
        N->next = L->current;
        if (L->current->prev == NULL) {
            L->front = N;
        } else {
            L->current->prev->next = N;
        }
        L->current->prev = N;
        L->length++;
      L->index++;
    }
}

/* insertAfterCurrent() - Inserts new element after current element.
   Pre: !isEmpty(), !offEnd(). */
void insertAfterCurrent(ListRef L, int data) {
   if ( L == NULL ) killProgram("Calling insertAfterCurrent() on NULL ListRef.");
    NodeRef N = newNode(data);
    if (isEmpty(L) || offEnd(L)) {
        killProgram("Method insertBeforeCurrent() failed to pass ppre isEmpty() &
offEnd() check.");
    } else {
        N->prev = L->current;
        N->next = L->current->next;
        if (N->next == NULL) {
            L->back = N;
        } else {
            N->next->prev = N;
        }
        L->current->next = N;
        L->length++;
```



```
    }
}

/* deleteFront() - Deletes front element.
   Pre: !isEmpty(). */
void deleteFront(ListRef L) {
   if ( L == NULL ) killProgram("Calling deleteFront() on NULL ListRef.");
   if (isEmpty(L)) {
        killProgram("Method deleteFront() cannot operate on an empty list.");
    } else {
      NodeRef temp = L->front;
        L->front = L->front->next;
        if (L->front == NULL)
            L->back = NULL;
        else
            L->front->prev = NULL;
      freeNode(&temp);
        L->length--;
    }
}

/* deleteBack() - Deletes back element.
   Pre: !isEmpty(). */
void deleteBack(ListRef L) {
   if ( L == NULL ) killProgram("Calling deleteBack() on NULL ListRef.");
   if ( isEmpty(L) )
      killProgram("Method deleteBack() cannot operate on an empty list.");
   else {
      NodeRef temp = L->back;
      L->back = L->back->prev;
      if ( L->back == NULL )
         L->front = NULL;
      else
         L->back->next = NULL;
      freeNode(&temp);
      L->length--;
   }
}

/* deleteCurrent() - Deletes current element.
   Pre: !isEmpty(), !offEnd()
   Post: offEnd() */
void deleteCurrent(ListRef L) {
   if ( L == NULL ) killProgram("Calling deleteCurrent() on NULL ListRef.");
    if ( isEmpty(L) || offEnd(L) ) {
```

```
            killProgram("Method deleteCurrent() cannot delete current node when list is
    empty or current is null.");
        }
        NodeRef temp = L->current;
        if ( L->current->prev == NULL ) {
            if ( L->current->next != NULL ) L->current->next->prev = NULL;
            L->front = L->current->next;
            if ( L->front == NULL ) L->back = NULL;
        } else if ( L->current->next == NULL ) {
            L->current->prev->next = NULL;
            L->back = L->current->prev;
        } else {
            L->current->prev->next = L->current->next;
            L->current->next->prev = L->current->prev;
        }
        L->current = NULL;
        L->index = -1;
        L->length--;
        freeNode(&temp);

        if (!offEnd(L)) {
            killProgram("Method deleteBack() failed to pass pre isEmpty() check.");
        }
    }


    /* Other functions ***************************************************
    *********************************************************************/

    /* copyList() - Returns a new list which is identical to this list. */
    ListRef copyList(ListRef L) {
       if ( L == NULL ) killProgram("Calling copyList() on NULL ListRef.");
       ListRef M = newList();
       NodeRef N;
       for ( N = L->front; N != NULL; N=N->next ){
          insertBack(M,N->data);
       }
       return M;
    }

    /* printList() prints current list to stdout */
    void printList(FILE* out, ListRef L) {
       if ( L == NULL ) killProgram("Calling printList() on NULL ListRef.");
       if ( isEmpty(L) ) printf("Nothing in List\n");
       else {
```



```
          NodeRef N = NULL;
          for ( N = L->front; N != NULL; N = N->next ){
             fprintf( out, "%d", N->data );
             if (N!=L->back) fprintf(out, " ");
          }
          /*fprintf(out, "\n");*/
       }
    }

    /* killProgram() - Utility method to report an error to user then exit. */
    void killProgram(char* error){
       printf("List.c: %s\n",error);
       exit(1);
    }
```

```
/* $Id: List.h,v 1.4 2011-10-08 19:09:41-07 - - $ */

/*
 * List.h
 * Header file for List.c
 * PA2
 * By B Stewart Bracken
 * bbracken@ucsc.edu
 * ID# 1187817
*/

#ifndef __LISTC_H__
#define __LISTC_H__

/* Useful typedefs */
typedef struct List* ListRef;
typedef enum {FALSE = 0, TRUE = 1} bool;


/* Constructor-Destructor for a List */
ListRef newList(void);
void freeList(ListRef* pL);

/* Access functions */
int getFront(ListRef L);
int getLength(ListRef L);
int isEmpty(ListRef L);
int getIndex(ListRef L);
int offEnd(ListRef L);
int getBack(ListRef L);
int getCurrent(ListRef L);
int equals(ListRef L, ListRef P);

/* Manipulation procedures */
void makeEmpty(ListRef L);
void moveTo(ListRef L, int i);
void movePrev(ListRef L);
void moveNext(ListRef L);
void insertFront(ListRef L, int data);
void insertBack(ListRef L, int data);
void insertBeforeCurrent(ListRef L, int data);
void insertAfterCurrent(ListRef L, int data);
void deleteFront(ListRef L);
void deleteBack(ListRef L);
void deleteCurrent(ListRef L);
```



```
void EnList(ListRef L, int data);
void DeList(ListRef L);

/* Other functions */
ListRef copyList(ListRef L);
void printList(FILE* out, ListRef L);
void killProgram(char* error);


#endif
```

```
#makefile for programming assignment 4



all: FindPath

FindPath: FindPath.o Graph.o List.o
    gcc -o FindPath FindPath.o Graph.o List.o

FindPath.o : FindPath.c Graph.h List.h
    gcc -c -ansi -Wall FindPath.c

GraphTest: GraphTest.o Graph.o List.o
    gcc -o GraphTest GraphTest.o Graph.o List.o

GraphTest.o : GraphTest.c Graph.h List.h
    gcc -c -ansi -Wall GraphTest.c

Graph.o : Graph.c Graph.h List.h
    gcc -c -ansi -Wall Graph.c

ListTest: ListTest.o List.o
    gcc -o ListTest ListTest.o List.o

ListTest.o : ListTest.c List.h
    gcc -c -ansi -Wall ListTest.c

List.o : List.c List.h
    gcc -c -ansi -Wall List.c

clean :
    rm -f FindPath GraphTest ListTest FindPath.o \
    GraphTest.o ListTest.o Graph.o List.o
```

```cpp
//
//  BinaryChop.cpp
//  Return index of element
//
//  Created by Stewart Bracken on 12/8/13.
//  Copyright (c) 2013 Stewart Bracken. All rights reserved.
//

#include <stdio.h>
#include <vector>
#include <thread>

#include "BinaryChop.h"
#include "FunctionalVector.h"

#include "MathUtil.h"

// My progression for better binary search functions.
// Chop6 should be the fastest.

//Iterative approach. Pretty fast, not as fast as chop2.
// O(nlogn) -- if lucky search hit, can return before nlogn.
int BinaryChop::chop1(int to_find, const std::vector<int>& data){
    int len = static_cast<int>(data.size()),
        low = 0;
    if(len == 0) return NOT_FOUND;
    int i, curr_data;

    while( len > 0 ){
        i = len/2 + low;
        curr_data = data[i];
        if( curr_data == to_find ){
            return i;
        }else if( to_find > curr_data ){
            //Need to search farther down array
            low = i+1;
            len = (len-1)/2;
        }else{ //to_find < curr_data
            //Search again at a smaller index.
            len = div_ceil( len-1, 2 );
        }
    }
    return NOT_FOUND;
}
```



```cpp
//Helper tail recursive function for chop2
static int chop2_rec(int& to_find, const std::vector<int>& data,
                     int& length, int& low){
    if(length == 0)
        return NOT_FOUND;
    int i = length/2 + low;
    int curr_data = data[i];
    if( curr_data == to_find ){
        return i;
    }else if( to_find > curr_data ){
        low = i+1;
        length = (length-1)/2;
    }else{
        length = div_ceil(length-1, 2);
    }
    return chop2_rec(to_find, data, length, low);
}

// Tail recursive chop.
// O(nlogn) -- if lucky search hit, can return before nlogn.
int BinaryChop::chop2(int to_find, const std::vector<int>& data){
    int length = static_cast<int>(data.size());
    int low = 0;
    return chop2_rec(to_find, data, length, low);
}


//Functional style array slicing binary chop
// O(nlogn) -- if lucky search hit, can return before nlogn.
int BinaryChop::chop3(int to_find, const std::vector<int>& data){
    FunctionalVector<int> fun_data(data); //copy data
    size_t i;
    int curr_data;
    while( fun_data.size() > 0 ){
        i = fun_data.size()/2;
        curr_data = fun_data[i];
        if( curr_data == to_find ){
            return static_cast<int>(fun_data.index_at(i));
        }else if( to_find > curr_data ){
            //Need to search farther down array
            fun_data.slice(i+1, (fun_data.size()-1)/2);
        }else{ //to_find < curr_data
            //Search again at a smaller index.
            fun_data.slice(0, div_ceil( static_cast<int>(fun_data.size()-1), 2) );
        }
```

```
        }
        return NOT_FOUND;
    }

    //thread function for doing work on data.
    void chop4_thread(int to_find, const std::vector<int> &data, int low, int length,
    int* result){
        int i = length/2 + low;
        int curr_data = data[i]; //thread-safe read

        if(curr_data == to_find){
            *result = i;
            return;
        }else if( to_find > curr_data ){
            low = i+1;
            length = (length-1)/2;
        }else{
            length = div_ceil(length-1, 2);
        }

        if(length == 0)
            return;

        std::thread continue_search(chop4_thread, to_find, data, low, length, result);
        continue_search.join();
    }

    void chop4_thread_spawn(int to_find, const std::vector<int> &data, int low, int
    length, int* result){
        if(length == 0) return;
        if(to_find < data[low] || to_find > data[low+length-1]) return;
        std::thread continue_search(chop4_thread, to_find, data, low, length, result);
        continue_search.join();
    }

    //very slow. binary search wan't really meant to be multithreaded.
    // O(nlogn) -- if lucky search hit, can return before nlogn.
    int BinaryChop::chop4( int to_find, const std::vector<int> &data ){
        int result = NOT_FOUND; //The thread which finds it sets result to idx
        int half = static_cast<int>(data.size())/2;
        std::thread left(chop4_thread_spawn, to_find, data, 0, half, &result);
        std::thread right(chop4_thread_spawn, to_find, data, half,
    div_ceil(static_cast<int>(data.size()), 2), &result);
        left.join();
        right.join();
```



```
        return result;
    }

    //Iterative approach 2. Attempting to be faster than tail recursion.
    // This ends up being just as fast as tail recursion.
    // Wow, it turns out I overengineered all my other solutions.
    // This one is much more simple and elegant. Nice dice.
    // θ(nlogn) -- bound above and below nlogn due to deferred equality.
    int BinaryChop::chop5( int to_find, const std::vector<int>& data ){
        int imax = static_cast<int>( data.size()-1 ),
            imin = 0,
            imid;
        if( imax < 0 ) return NOT_FOUND;

        while( imin < imax ){
            imid = (imin + imax)/2;
            if( to_find > data[imid] ){
                imin = imid+1;
            }else{ //to_find < curr_data
                imax = imid;
            }
        }
        if( imin == imax && data[imin] == to_find ){
            return imin;
        }
        return NOT_FOUND;
    }

    //Helper tail recursive function for chop6
    static int chop6_rec( int& to_find, const std::vector<int>& data,
                          int& imin, int& imax ){
        if( imin == imax ){
            if( data[imin] == to_find ){
                return imin;
            } else {
                return NOT_FOUND;
            }
        }

        int imid = ( imin + imax )/2;

        if( to_find > data[imid] ){
            imin = imid+1;
        }else{
            imax = imid;
```

```
        }

        return chop6_rec( to_find, data, imin, imax );
    }

    // Θ(nlogn) -- bound above and below nlogn due to deferred equality.
    int BinaryChop::chop6( int to_find, const std::vector<int>& data ){
        int imax = static_cast<int>( data.size()-1 );
        int imin = 0;
        if( imax < 0 )
            return NOT_FOUND;
        return chop6_rec( to_find, data, imin, imax );
    }
```

```
//
//  BinaryChop.h
//  CppTests
//
//  Created by Stewart Bracken on 12/8/13.
//  Copyright (c) 2013 Stewart Bracken. All rights reserved.
//

#ifndef InterviewTests_BinaryChop_h
#define InterviewTests_BinaryChop_h

#include <vector>

#define NOT_FOUND -1

//These all should return the same value if supplied equivalent args
namespace BinaryChop {
    //Iterative - pretty fast!
    int chop1(int to_find, const std::vector<int>& data);
    //Tail Recursive - fast!
    int chop2(int to_find, const std::vector<int>& data);
    //vector slicing - slow due to data copying.
    int chop3(int to_find, const std::vector<int>& data);
    //concurrent searching - slow
    int chop4(int to_find, const std::vector<int>& data);
    //iterative with deferred equality
    int chop5(int to_find, const std::vector<int>& data);
    //attempt to speed up recursive chop
    int chop6(int to_find, const std::vector<int>& data);
}

#endif
```

```
 //
//  BinaryChop_unitttest.cpp
//  CppTests
//
//  Created by Stewart Bracken on 12/9/13.
//  Copyright (c) 2013 Stewart Bracken. All rights reserved.
//

#include <iomanip>
#include <iostream>
#include <vector>
#include <gtest/gtest.h>

#include <time.h>

double get_seconds(clock_t ct)
{
    return ((double)ct)/CLOCKS_PER_SEC;
}

#include "BinaryChop.h"

typedef int (*chop_ptr)(int, const std::vector<int>&);

TEST(BinaryChop, Chop_iterative){
    clock_t start = clock();

    //Use point because it's easier to copy paste test data
    chop_ptr chop = &(BinaryChop::chop1);
    for(int i=0;i<1000;++i){
    std::vector<int> data = {  };
    ASSERT_TRUE( NOT_FOUND == chop(3, data));
    data = {1};
    ASSERT_TRUE(NOT_FOUND == chop(3, data));
    ASSERT_TRUE(0 == chop(1, data));

    data = {1, 3, 5};
    ASSERT_TRUE(0 == chop(1, data));
    ASSERT_TRUE(1 == chop(3, data));
    ASSERT_TRUE(2 == chop(5, data));
    ASSERT_TRUE(NOT_FOUND == chop(0, data));
    ASSERT_TRUE(NOT_FOUND == chop(2, data));
    ASSERT_TRUE(NOT_FOUND == chop(4, data));
    ASSERT_TRUE(NOT_FOUND == chop(6, data));
```



```
    data = {1, 3, 5, 7};
    ASSERT_TRUE(0 == chop(1, data));
    ASSERT_TRUE(1 == chop(3, data));
    ASSERT_TRUE(2 == chop(5, data));
    ASSERT_TRUE(3 == chop(7, data));
    ASSERT_TRUE(NOT_FOUND == chop(0, data));
    ASSERT_TRUE(NOT_FOUND == chop(2, data));
    ASSERT_TRUE(NOT_FOUND == chop(4, data));
    ASSERT_TRUE(NOT_FOUND == chop(6, data));
    ASSERT_TRUE(NOT_FOUND == chop(8, data));

    for(int i = 0; i < 500; ++i){
        data.push_back(i*2 + 9);
    }
    ASSERT_TRUE(250 == chop(501, data));
    ASSERT_TRUE(500 == chop(1001, data));
    ASSERT_TRUE(NOT_FOUND == chop(1000000, data));
    }
    clock_t end = clock();
    std::cout<<std::setprecision(10);
    std::cout<<"("<<get_seconds(end-start)<<" seconds)"<<std::endl;
}



TEST(BinaryChop, Chop_recursive){
    clock_t start = clock();
    chop_ptr chop = &BinaryChop::chop2;
    for(int i=0;i<1000;++i){
    std::vector<int> data = {  };
    ASSERT_TRUE( NOT_FOUND == chop(3, data));
    data = {1};
    ASSERT_TRUE(NOT_FOUND == chop(3, data));
    ASSERT_TRUE(0 == chop(1, data));

    data = {1, 3, 5};
    ASSERT_TRUE(0 == chop(1, data));
    ASSERT_TRUE(1 == chop(3, data));
    ASSERT_TRUE(2 == chop(5, data));
    ASSERT_TRUE(NOT_FOUND == chop(0, data));
    ASSERT_TRUE(NOT_FOUND == chop(2, data));
    ASSERT_TRUE(NOT_FOUND == chop(4, data));
    ASSERT_TRUE(NOT_FOUND == chop(6, data));
```

```
        data = {1, 3, 5, 7};
        ASSERT_TRUE(0 == chop(1, data));
        ASSERT_TRUE(1 == chop(3, data));
        ASSERT_TRUE(2 == chop(5, data));
        ASSERT_TRUE(3 == chop(7, data));
        ASSERT_TRUE(NOT_FOUND == chop(0, data));
        ASSERT_TRUE(NOT_FOUND == chop(2, data));
        ASSERT_TRUE(NOT_FOUND == chop(4, data));
        ASSERT_TRUE(NOT_FOUND == chop(6, data));
        ASSERT_TRUE(NOT_FOUND == chop(8, data));

        for(int i = 0; i < 500; ++i){
            data.push_back(i*2 + 9);
        }
        ASSERT_TRUE(250 == chop(501, data));
        ASSERT_TRUE(500 == chop(1001, data));
        ASSERT_TRUE(NOT_FOUND == chop(1000000, data));
        }
        clock_t end = clock();
        std::cout<<std::setprecision(10);
        std::cout<<"("<<get_seconds(end-start)<<" seconds)"<<std::endl;
    }



    TEST(BinaryChop, Chop_functional_vector){
        clock_t start = clock();
        chop_ptr chop = &BinaryChop::chop3;
        for(int i=0;i<1000;++i){
        std::vector<int> data = {  };
        ASSERT_TRUE( NOT_FOUND == chop(3, data));
        data = {1};
        ASSERT_TRUE(NOT_FOUND == chop(3, data));
        ASSERT_TRUE(0 == chop(1, data));

        data = {1, 3, 5};
        ASSERT_TRUE(0 == chop(1, data));
        ASSERT_TRUE(1 == chop(3, data));
        ASSERT_TRUE(2 == chop(5, data));
        ASSERT_TRUE(NOT_FOUND == chop(0, data));
        ASSERT_TRUE(NOT_FOUND == chop(2, data));
        ASSERT_TRUE(NOT_FOUND == chop(4, data));
        ASSERT_TRUE(NOT_FOUND == chop(6, data));

        data = {1, 3, 5, 7};
```



```
        ASSERT_TRUE(0 == chop(1, data));
        ASSERT_TRUE(1 == chop(3, data));
        ASSERT_TRUE(2 == chop(5, data));
        ASSERT_TRUE(3 == chop(7, data));
        ASSERT_TRUE(NOT_FOUND == chop(0, data));
        ASSERT_TRUE(NOT_FOUND == chop(2, data));
        ASSERT_TRUE(NOT_FOUND == chop(4, data));
        ASSERT_TRUE(NOT_FOUND == chop(6, data));
        ASSERT_TRUE(NOT_FOUND == chop(8, data));

        for(int i = 0; i < 500; ++i){
            data.push_back(i*2 + 9);
        }
        ASSERT_TRUE(250 == chop(501, data));
        ASSERT_TRUE(500 == chop(1001, data));
        ASSERT_TRUE(NOT_FOUND == chop(1000000, data));
        }
        clock_t end = clock();
        std::cout<<std::setprecision(10);
        std::cout<<"("<<get_seconds(end-start)<<" seconds)"<<std::endl;
    }


    TEST(BinaryChop, Chop_concurrent){
        clock_t start = clock();
        chop_ptr chop = &BinaryChop::chop4;
        for(int i=0;i<1000;++i){
        std::vector<int> data = {  };
        ASSERT_TRUE( NOT_FOUND == chop(3, data));
        data = {1};
        ASSERT_TRUE(NOT_FOUND == chop(3, data));
        ASSERT_TRUE(0 == chop(1, data));

        data = {1, 3, 5};
        ASSERT_TRUE(0 == chop(1, data));
        ASSERT_TRUE(1 == chop(3, data));
        ASSERT_TRUE(2 == chop(5, data));
        ASSERT_TRUE(NOT_FOUND == chop(0, data));
        ASSERT_TRUE(NOT_FOUND == chop(2, data));
        ASSERT_TRUE(NOT_FOUND == chop(4, data));
        ASSERT_TRUE(NOT_FOUND == chop(6, data));

        data = {1, 3, 5, 7};
        ASSERT_TRUE(0 == chop(1, data));
        ASSERT_TRUE(1 == chop(3, data));
```

```
        ASSERT_TRUE(2 == chop(5, data));
        ASSERT_TRUE(3 == chop(7, data));
        ASSERT_TRUE(NOT_FOUND == chop(0, data));
        ASSERT_TRUE(NOT_FOUND == chop(2, data));
        ASSERT_TRUE(NOT_FOUND == chop(4, data));
        ASSERT_TRUE(NOT_FOUND == chop(6, data));
        ASSERT_TRUE(NOT_FOUND == chop(8, data));

        for(int i = 0; i < 500; ++i){
            data.push_back(i*2 + 9);
        }
        ASSERT_TRUE(250 == chop(501, data));
        ASSERT_TRUE(500 == chop(1001, data));
        ASSERT_TRUE(NOT_FOUND == chop(1000000, data));
        }
        clock_t end = clock();
        std::cout<<std::setprecision(10);
        std::cout<<"("<<get_seconds(end-start)<<" seconds)"<<std::endl;
    }

    TEST(BinaryChop, Chop_deferred_equality){
        clock_t start = clock();
        chop_ptr chop = &BinaryChop::chop5;
        for(int i=0;i<1000;++i){
        std::vector<int> data = {  };
        ASSERT_TRUE( NOT_FOUND == chop(3, data));
        data = {1};
        ASSERT_TRUE(NOT_FOUND == chop(3, data));
        ASSERT_TRUE(0 == chop(1, data));

        data = {1, 3, 5};
        ASSERT_TRUE(0 == chop(1, data));
        ASSERT_TRUE(1 == chop(3, data));
        ASSERT_TRUE(2 == chop(5, data));
        ASSERT_TRUE(NOT_FOUND == chop(0, data));
        ASSERT_TRUE(NOT_FOUND == chop(2, data));
        ASSERT_TRUE(NOT_FOUND == chop(4, data));
        ASSERT_TRUE(NOT_FOUND == chop(6, data));

        data = {1, 3, 5, 7};
        ASSERT_TRUE(0 == chop(1, data));
        ASSERT_TRUE(1 == chop(3, data));
        ASSERT_TRUE(2 == chop(5, data));
        ASSERT_TRUE(3 == chop(7, data));
        ASSERT_TRUE(NOT_FOUND == chop(0, data));
```



```
        ASSERT_TRUE(NOT_FOUND == chop(2, data));
        ASSERT_TRUE(NOT_FOUND == chop(4, data));
        ASSERT_TRUE(NOT_FOUND == chop(6, data));
        ASSERT_TRUE(NOT_FOUND == chop(8, data));

        for(int i = 0; i < 500; ++i){
            data.push_back(i*2 + 9);
        }
        ASSERT_TRUE(250 == chop(501, data));
        ASSERT_TRUE(500 == chop(1001, data));
        ASSERT_TRUE(NOT_FOUND == chop(1000000, data));
        }
        clock_t end = clock();
        std::cout<<std::setprecision(10);
        std::cout<<"("<<get_seconds(end-start)<<" seconds)"<<std::endl;
    }

    TEST(BinaryChop, recursive_speedup){
        clock_t start = clock();
        chop_ptr chop = &BinaryChop::chop6;
        for(int i=0;i<1000;++i){
        std::vector<int> data = {  };
        ASSERT_TRUE( NOT_FOUND == chop(3, data));
        data = {1};
        ASSERT_TRUE(NOT_FOUND == chop(3, data));
        ASSERT_TRUE(0 == chop(1, data));

        data = {1, 3, 5};
        ASSERT_TRUE(0 == chop(1, data));
        ASSERT_TRUE(1 == chop(3, data));
        ASSERT_TRUE(2 == chop(5, data));
        ASSERT_TRUE(NOT_FOUND == chop(0, data));
        ASSERT_TRUE(NOT_FOUND == chop(2, data));
        ASSERT_TRUE(NOT_FOUND == chop(4, data));
        ASSERT_TRUE(NOT_FOUND == chop(6, data));

        data = {1, 3, 5, 7};
        ASSERT_TRUE(0 == chop(1, data));
        ASSERT_TRUE(1 == chop(3, data));
        ASSERT_TRUE(2 == chop(5, data));
        ASSERT_TRUE(3 == chop(7, data));
        ASSERT_TRUE(NOT_FOUND == chop(0, data));
        ASSERT_TRUE(NOT_FOUND == chop(2, data));
        ASSERT_TRUE(NOT_FOUND == chop(4, data));
        ASSERT_TRUE(NOT_FOUND == chop(6, data));
```

```
        ASSERT_TRUE(NOT_FOUND == chop(8, data));

        for(int i = 0; i < 500; ++i){
            data.push_back(i*2 + 9);
        }
        ASSERT_TRUE(250 == chop(501, data));
        ASSERT_TRUE(500 == chop(1001, data));
        ASSERT_TRUE(NOT_FOUND == chop(1000000, data));
        }
        clock_t end = clock();
        std::cout<<std::setprecision(10);
        std::cout<<"("<<get_seconds(end-start)<<" seconds)"<<std::endl;
    }
```

```cpp
//
//  FunctionalVector_unittest.cpp
//  CppPractice
//
//  Created by Stewart Bracken on 12/12/13.
//  Copyright (c) 2013 Stewart Bracken. All rights reserved.
//

#include <stdio.h>
#include <gtest/gtest.h>
#include "FunctionalVector.h"

TEST(FunctionalVector, comprehensive_test){
    FunctionalVector<int> original({1,2,3,4,5});
    FunctionalVector<int> sliced = original;
    sliced.slice(1,3); //slice to indices 1->3s
    ASSERT_EQ(original[1], sliced[0]);
    sliced[0] = 10;
    original[0] = 11;
    std::cout<<"original: "<<original<<std::endl;
    std::cout<<"sliced: "<<sliced<<std::endl;
    FunctionalVector<int> slice_copy = original.slice(2, 4);
    std::cout<<"original: "<<original<<std::endl;
    std::cout<<"slice_copy: "<<slice_copy<<std::endl;
}
```

```
//
//  FunctionalVector.h
//  CppPractice
//
//  Created by Stewart Bracken on 12/10/13.
//  Copyright (c) 2013 Stewart Bracken. All rights reserved.
//

#ifndef __CppPractice__FunctionalVector__
#define __CppPractice__FunctionalVector__

#include <iostream>
#include <vector>
#include <algorithm> // for copy
#include <iterator> // for ostream_iterator

//Vector wrapper that can 'slice' a vector to specific range offset.
// for example a vector {1,2,3}.slice(1,2) = {2,3} and indexing to 0 returns 2
// instead of 1.
template <class T>
class FunctionalVector : public std::vector<T> {
    size_t i_start, i_length, i_end;
public:
    FunctionalVector();

    FunctionalVector(const FunctionalVector<T>& rhs):
std::vector<T>(static_cast<std::vector<T> >(rhs)),
    i_start(rhs.i_start), i_end(rhs.i_end),
i_length(rhs.i_length){/*std::cout<<"fun_vec copy ctor"<<std::endl;*/}

    //Copy data constructor
    FunctionalVector(const std::vector<T>& other): std::vector<T>(other),
    i_start(0), i_end(other.size()), i_length(other.size()){/*std::cout<<"vec copy
ctor"<<std::endl;*/}

    FunctionalVector<T>& slice(size_t new_start, size_t new_length){
        i_start += new_start;
        i_length = new_length;
        i_end = i_start +i_length;
        return *this;
    }

    //Override vector's bracket operator to place modified indices.
    T& operator[](size_t idx){
        if(idx >= i_length && i_length > 0)
```



```
            idx %= i_length;
        return std::vector<T>::operator[](index_at(idx));
    }
    const T& operator[](size_t idx)const{
        if(idx >= i_length && i_length > 0)
            idx %= i_length;
        return std::vector<T>::operator[](index_at(idx));
    }

    void reset(){i_start=0; i_length=std::vector<T>::size(); i_end = i_length;}


    size_t index_at(size_t norm_idx)const{
        return i_start + norm_idx;
    }

    size_t size()const{
        return i_length;
    }
    friend std::ostream& operator<<(std::ostream& out, const FunctionalVector& me){
        copy(me.begin() + me.i_start, me.begin() + (me.i_end),
std::ostream_iterator<T>(out, ", "));
        return out;
    }
};

#endif /* defined(__CppPractice__FunctionalVector__) */
```

```cpp
//
//  MathUtil.cpp
//  CppTests
//
//  Created by Stewart Bracken on 12/9/13.
//  Copyright (c) 2013 Stewart Bracken. All rights reserved.
//

//
//  MathUtil.h
//  CppTests
//
//  Created by Stewart Bracken on 12/9/13.
//  Copyright (c) 2013 Stewart Bracken. All rights reserved.
//

#ifndef __InterviewTests__MathUtil__
#define __InterviewTests__MathUtil__

#include <iostream>

int div_ceil(int dividend, int divisor);

#endif /* defined(__InterviewTests__MathUtil__) */


//This function has possible overflow from dividend + divisor
static int div_ceil_overflow(int dividend, int divisor){
    return (dividend + divisor - 1) / divisor;
}

int div_ceil(int dividend, int divisor){
    if(dividend == 0)
        return div_ceil_overflow(dividend, divisor);
    return 1 + ((dividend - 1) / divisor); // if x != 0
}
```

```
//
//  bitcpy.cpp
//  FreeRangeInterviewTest
//
//  Created by Stewart Bracken on 2/5/14.
//  Copyright (c) 2014 Stewart Bracken. All rights reserved.
//

//
//  bitcpy.h
//  FreeRangeInterviewTest
//
//  Created by Stewart Bracken on 2/5/14.
//  Copyright (c) 2014 Stewart Bracken. All rights reserved.
//

#ifndef __FreeRangeInterviewTest__bitcpy__
#define __FreeRangeInterviewTest__bitcpy__

#include <iostream>

void bitcpy(void* src, void* dst, int numBits);

#endif /* defined(__FreeRangeInterviewTest__bitcpy__) */



void bitcpy(void* src, void* dst, int numBits){
    char* srcC = static_cast<char*>(src);
    char* dstC = static_cast<char*>(dst);
    // Copy as many bytes as possible
    while (numBits >= 8){
        numBits -= 8;
        *dstC = *static_cast<char*>(srcC); //copy 1 byte at a time into dst
        srcC++;
        dstC++;
    }
    // Copy the remianing bits using bit operators
    if (numBits > 0 ){
        char mask = 1 << 7; //with a one in the most significant bit position
        while (numBits > 0){
            char result = mask & *srcC;
            if ( result )
                *dstC |= result; //set it
            else
```



```
                *dstC &= ~(mask); //clear it
            numBits--;
            mask >>= 1;
        }
    }
}
```

```
//
//  Connect4Evaulator.cpp
//  FreeRangeInterviewTest
//
//  Created by Stewart Bracken on 2/6/14.
//  Copyright (c) 2014 Stewart Bracken. All rights reserved.
//

#include <iostream>

//
//  Connect4.hpp
//  FreeRangeInterviewTest
//
//  Created by Stewart Bracken on 2/6/14.
//  Copyright (c) 2014 Stewart Bracken. All rights reserved.
//

#ifndef FreeRangeInterviewTest_Connect4Evaulator_hpp
#define FreeRangeInterviewTest_Connect4Evaulator_hpp

#include <vector>

typedef std::vector<char> conn4grid;

//Name of the game!
const int connect = 4;

const char red ='R',
           black = 'B',
           empty = '.';

class Connect4Evaulator {
    const int GRID_SIZE;

//****** PRIVATE METHODS *******//
private:
    int get_index (int x, int y, int _width);

    // Checks a tile against the next 3 using vx/vy as the direction.
    // Returns true if it's found a connect 4.
    bool has_connection4 (const conn4grid& connect4grid, char curr, int curr_x, int
curr_y, int vx, int vy, int width);

    //PRE: out_grid is empty
```



```
    void transpose (const conn4grid& grid, conn4grid& out_grid, int& width, int&
height);

    // Mirror flip rows
    void exchange_rows (conn4grid& grid, int width, int height);

    //Mirror flip columns
    void exchange_columns (conn4grid& grid, int width, int height);

    // Push all non-empty spaces downwards (increasing y)
    void apply_gravity (conn4grid& grid, int width, int height);


public:

    Connect4Evaulator(int grid_size = 42):GRID_SIZE(grid_size){}


//****** RETURN STATES *******//
    enum { RED_WIN, RED_LOSE, DRAW, UNFINISHED, NEITHER, LEFT, RIGHT, ERROR };


//****** PUBLIC METHODS *******//
    void print_grid (const conn4grid& grid, int width, int height);

    // Returns RED_WIN, RED_LOSE, DRAW, UNFINISHED, or ERROR
    int evaluate_conn4_state(const conn4grid& connect4grid, int width = 7, int
height = 6 );

    // Returns LEFT, RIGHT, NEITHER, or ERROR
    int evaluate_rolled_conn4_state (const conn4grid& original_grid, int width = 7,
int height = 6);


};


#endif


int Connect4Evaulator::get_index (int x, int y, int _width){
    return y * _width + x;
}

bool Connect4Evaulator::has_connection4 (const conn4grid& connect4grid,
```

```
                                        char curr, int curr_x, int curr_y,
                                        int vx, int vy, int width){
    for(int i=0; i < 3; ++i){
        curr_x += vx;
        curr_y += vy;
        if (curr != connect4grid[get_index(curr_x, curr_y, width)])
            return false;
    }
    return true;
}

void Connect4Evaulator::transpose (const conn4grid& grid, conn4grid& out_grid,
                                   int& width, int& height){
    //transpose by swapping rows with columns.
    for (int x = 0; x < width; ++x){
        for (int y = 0; y < height; ++y){
            out_grid.push_back (grid[get_index (x,y,width)] );
        }
    }
    std::swap (width, height);
}

void Connect4Evaulator::exchange_rows (conn4grid& grid, int width, int height){
    int half = height/2;
    for(int i=0; i< half; ++i){
        int row1 = i * width,
        row2 = (height - i - 1) * width;
        for (int x = 0; x < width; ++x, ++row1, ++row2){
            std::swap( grid[row1], grid[row2] );
        }
    }
}

void Connect4Evaulator::exchange_columns (conn4grid& grid, int width,
                                          int height){
    int half = width/2;
    for(int i=0; i< half; ++i){
        int col1 = i,
        col2 = width-1-i;
        for (int x = 0; x < width; ++x, col1+=width, col2+=width){
            std::swap( grid[col1], grid[col2] );
        }
    }
}
```



```
void Connect4Evaulator::apply_gravity (conn4grid& grid, int width, int height){
    const int NONE = -1;
    int last_empty_idx = NONE;
    int i;
    for (int x = 0; x < width; ++x){
        last_empty_idx = NONE;
        for (int y = height-1; y >= 0; --y){
            i = get_index (x, y, width);
            if (grid[i] == empty){
                if (last_empty_idx == NONE)
                    last_empty_idx = i;
            }else if (last_empty_idx  > NONE ) {
                //we've found an empty before and have a red or blue here
                std::swap( grid[i], grid[last_empty_idx] );
                last_empty_idx -= width; //up a row
            }
        }
    }
}


void Connect4Evaulator::print_grid (const conn4grid& grid, int width,
                                    int height){
    for(int y=0; y<height; ++y){
        for (int x=0; x<width; ++x){
            std::cout << grid[y*width+x] << " ";
        }
        std::cout<<std::endl;
    }
}

int Connect4Evaulator::evaluate_conn4_state(const conn4grid& connect4grid,
                                            int width, int height ){
    if (connect4grid.size() != GRID_SIZE)
        return ERROR;

    const int width_check_max = width - (width-connect)+1, //5 on standard grid
       height_check_max = height - (height - connect) + 1; // 4   "  "        "

    int result = DRAW;

    // Loop variables
    char curr; int i; bool has_connected_four;
    for(int y=0; y < height; ++y){
        for( int x = 0; x < width; ++x){
```

```
            i = get_index(x, y, width);
            curr = connect4grid[i];

            if ( curr == empty ){
                result = UNFINISHED;
                continue;
            }

            has_connected_four = false;

            if ( x < width_check_max ){
                //check across horz for a connect 4
                has_connected_four = has_connection4(connect4grid, curr,
                                                     x, y, 1, 0, width);
            }
            if (!has_connected_four)
                if ( y < height_check_max ){
                    //check down vertically for a connect 4
                    has_connected_four = has_connection4(connect4grid, curr,
                                                         x, y, 0, 1, width);
                }
            if (!has_connected_four)
                if ( x < width_check_max && y < height_check_max ){
                    //check diagonally down & to right for connect 4
                    has_connected_four = has_connection4(connect4grid, curr,
                                                         x, y, 1, 1, width);
                }
            if (!has_connected_four)
                if( y < height_check_max && x >= connect-1 ){
                    //check diagonally down & to left to c4.
                    has_connected_four = has_connection4(connect4grid, curr,
                                                         x, y, -1, 1, width);
                }

            if ( has_connected_four ){
                if ( curr == red )
                    return RED_WIN;
                else
                    return RED_LOSE;
            }
        }
    }

    return result;
}
```



```
int Connect4Evaulator::evaluate_rolled_conn4_state (
                                                    const conn4grid& original_grid,
                                                    int width, int height){
    if (original_grid.size() != GRID_SIZE)
        return ERROR;
    // Rotate 90 left == counter clockwise
    // Rotate 90 right == clockwise

    // To rotate first we transpose, by exhanging rows with columns.
    conn4grid trans;
    transpose (original_grid, trans, width, height);

    // Mirror flip rows at middle to rotate left.
    conn4grid left ( trans ); // Copy transposed so we can modify in place.
    exchange_rows(left, width, height);

    apply_gravity (left, width, height);
    int left_result = evaluate_conn4_state(left, width, height);
    if ( left_result == RED_WIN || left_result == RED_LOSE ){
        return LEFT;
    }

    // Mirror flip columns to rotate right.
    conn4grid right;
    std::swap(right, trans); // Don't need trans anymore so just swap it.
    exchange_columns(right, width, height);

    apply_gravity (right, width, height);
    int right_result = evaluate_conn4_state(right, width, height);
    if ( right_result == RED_WIN || right_result == RED_LOSE ){
        return RIGHT;
    }

    return NEITHER;
}
```

```cpp
//  Find the expression in a string of digits that evaluate
//  to a particular answer
//  Created by Stewart Bracken on 11/13/13.
//  Copyright (c) 2013 Stewart Bracken. All rights reserved.
//

#include <iostream>
#include <unordered_map> //lookup table
#include <vector>
#include <sstream> //splitting strings

#include <string>

using namespace std;

void doScript(const string& script){

}

void yaddida(){
    vector<string> s;
    const vector<string>* scripts = &s;
    doScript(scripts->at(0));
}

/*******************************
 start test question 1
 *******************************/

//a string of an expression and it's evaluated numeric answer
typedef unordered_map<string, int> expr_lookup;

// split a string at every occurance of delim
// @param s - the string to split
// @param delim - the delimiter
// @param (optional) elems - the vector to push the delimited strings into
vector<string>& split(const string &s, char delim, vector<string>& elems) {
    stringstream ss(s);
    string item;
    while (getline(ss, item, delim)) {
        elems.push_back(item);
    }
    return elems;
}
```



```cpp
vector<string> split(const string &s, char delim) {
    vector<string> elems;
    split(s, delim, elems);
    return elems;
}

template <typename T>
T StringToNumber ( const string &Text ){
    stringstream ss(Text);
    T result;
    return ss >> result ? result : 0;
}

// Evaluate a string expression.
// @param expr - a string expression with digits, +, or *
// @param lookup_table - an unordered_map<string, int>
// @return - 0 if blank string, or the answer
int evaluate(string expr, expr_lookup& lookup_table){
    auto lookup = lookup_table.find(expr);
    if (lookup != lookup_table.end()){
        return lookup->second; //already know solution
    }
    //actually evaluate it
    vector<string> plus = split(expr,'+'), mult;
    int out = 0;
    for(int i=0; i<plus.size();++i){
        mult.clear();
        mult = split(plus[i],'*');
        int m = StringToNumber<int>(mult[0]);
        for(int j=1;j<mult.size();++j){
            m *= StringToNumber<int>(mult[j]);
        }
        out += m;
    }
    lookup_table.insert({expr,out}); //SAVE ANSWER
    return out;
}


// Recursively try to evaluate combinations of digits
// @param expr_so_far - supply a blank string initially.
//                      It must have a + or * at the end or be blank.
// @param digits - pass in the initial digits string
// @param answer - the desired answer
// @param lookup_table - initally supply an empty expr_lookup table
```

```
// @return - the desired expression or an empty string if not found.
//expr_so_far already has a + or * at the end, or blank
string find_expression_rec(string& expr_so_far, string& digits, const int answer,
expr_lookup& lookup_table){
    for( int i = 1; i < digits.size(); ++i){
        string l = digits.substr(0, i); //new digits
        string r = digits.substr(i);  //new digits
        string plus = expr_so_far + l + "+";
        string mult = expr_so_far + l + "*";
        if (evaluate(plus + r, lookup_table) == answer) return plus + r;
        if (evaluate(mult + r, lookup_table) == answer) return mult + r;
        string out = find_expression_rec(plus, r, answer, lookup_table);
        if ( out != "" ) return out;
        out = find_expression_rec(mult, r, answer, lookup_table);
        if( out != "" ) return out;
    }
    return "";
}

// Attempt to find an expression by adding +/* within the digits string to
// evaluate to answer. Brute force :(
// @param digits - a string of digits
// @param answer - integer which you desire to find an expression for
// @return - either the expression or "no solution"
string find_expression(string digits, int answer){
    const int a = answer;
    string d = digits;
    string expr = "";
    expr_lookup lookup_table;
    string out = find_expression_rec(expr,d,a,lookup_table);
    if ( out == "" ) out = "no solution";
    return out;
}

int main(int argc, const char * argv[])
{
    string test1 = find_expression( "1231231234", 11353 );
    cout << test1 <<endl;

    string test2 = find_expression( "3456237490", 1185 );
    cout << test2 <<endl;

    string test3 = find_expression( "3456237490", 9191 );
    cout << test3 <<endl;
```



```
    return 0;
}
```

```
//
//  List.cpp
//  CppPractice
//
//  Created by Stewart Bracken on 12/9/13.
//  Copyright (c) 2013 Stewart Bracken. All rights reserved.
//


#include <stdio.h> //NULL
#include <iostream>

//
//  Very simple list class for practicing list algorithms, not intended for
//  production.
//
//  Created by Stewart Bracken on 12/4/13.
//  Copyright (c) 2013 Stewart Bracken. All rights reserved.
//

#ifndef InterviewPractice_List_h
#define InterviewPractice_List_h

template <class T>
class List {
    template <class U>
    class Node{
        friend class List<T>;
        T data;
        Node<U>* next;
        Node(U _data):data(_data), next(0){
        }
    };
    typedef Node<T> Node_t;
    Node_t* head;
    Node_t* tail;
public:
    List() : head(0),tail(0) {};
    ~List<T>();
    void insert_back(T data);
    void insert_front(T data);
    //Skip k, then reverse k. Repeat until end.
    void reverse_k(T k);
    void print() const;
};
```



```
#endif


template <class T>
List<T>::~List(){
    Node_t* curr = head;
    while(curr != NULL){
        head = curr->next;
        delete curr;
        curr = head;
    }
}

template <class T>
void List<T>::insert_back(T data){
    Node_t* new_node =new Node_t(data);
    if(tail)
        tail->next = new_node;
    tail =  new_node;
    if(head == 0){
            head = tail;
    }
}

template <class T>
void List<T>::insert_front(T data){
        Node_t* tmp = new Node_t(data);
        tmp->next = head;
        if(head==NULL){tail = tmp; tmp->next=NULL;}
        head = tmp;
    }


template <class T>
void List<T>::reverse_k(T k){
    Node_t* curr = head;
    Node_t* tmp;
    while( curr != NULL ){
        //skip up to k nodes
        for(int i=0; i<k && curr != NULL; ++i){
            curr = curr->next;
        }

        //reverse up to k nodes
```

```
            tmp = curr;
            List<T> reverse;
            for(int i=0; i<k && curr != NULL; ++i){
                reverse.insert_front(curr->data);
                curr = curr->next;
            }
            curr = tmp;
            tmp = reverse.head;
            for(int i=0; i<k && tmp != NULL; ++i){
                curr->data = tmp->data;
                curr = curr->next;
                tmp = tmp->next;
            }
        }
    }

    template<class T>
    void List<T>::print() const{
        for( Node_t* itor = head; itor != 0; itor = itor->next){
            std::cout << itor->data << ", ";
        }
    }

    //
    //  ListKTest_unittest.cpp
    //  CppTests
    //
    //  Created by Stewart Bracken on 12/5/13.
    //  Copyright (c) 2013 Stewart Bracken. All rights reserved.
    //

    #include <gtest/gtest.h>

    #include "List.h"
    //include the template implementation
    #include "List.cpp"

    TEST(ListKTest, SkipKReverseK){
        List<int> list;
        for(int i = 0; i < 20; ++i){
            list.insert_back(i+1);
        }
        list.print();
        list.reverse_k(20);
        list.print();
```



```
    }
```

```
/*
 * Stewart Bracken Copyright 2014
 * ofApp.cpp
 */


/*
 * Stewart Bracken Copyright 2014
 * OFAPP.H
 */
#pragma once

#include "ofMain.h"

#include "ofxLua.h"
#include "ofxLuaBindings.h" // the OF api -> lua binding

#include "ofxUI.h"

#include <map>

class ofApp : public ofBaseApp, ofxLuaListener {

public:

    // main
    void setup();
    void update();
    void draw();
    void exit();


    // input
    void keyPressed(int key);
    void mouseMoved(int x, int y );
    void mouseDragged(int x, int y, int button);
    void mousePressed(int x, int y, int button);
    void mouseReleased(int x, int y, int button);

    // ofxLua error callback
    void errorReceived(string& msg);

    // script control
    void reloadScript();
```



```
    ofxLua lua;
    vector<string> scripts;
    int currentScript;
    bool hasError;
    std::string error;

    ofxUICanvas *gui;
    void guiEvent(ofxUIEventArgs &e);
    ofxUICanvas *guiConsole;
    void guiConsoleEvent(ofxUIEventArgs &e);

    void addConsoleMessage(const string&);

private:
    void build_directory_gui();
    void add_to_gui(string path);
    map<string,string> directory_map;
    void reset_directory_gui();
};


class ofGUILoggerChannel: public ofBaseLoggerChannel{
public:
    ofGUILoggerChannel(ofApp* _app):app(_app){};
    //virtual ~ofGUILoggerChannel(){};
    void log(ofLogLevel level, const string & module, const string & message);
    void log(ofLogLevel level, const string & module, const char* format, ...);
    void log(ofLogLevel level, const string & module, const char* format, va_list
args);
private:
    ofApp* app;
};


//--------------------------------------------------------------
void ofApp::setup() {

    ofSetVerticalSync(true);
    ofSetLogLevel("ofxLua", OF_LOG_NOTICE);
    ofSetEscapeQuitsApp(false);

    hasError= false;

    //file browser gui
    gui = new ofxUICanvas();
```

```
        ofAddListener(gui->newGUIEvent,this,&ofApp::guiEvent);

        //console gui
        guiConsole = new ofxUICanvas();
        ofAddListener(gui->newGUIEvent,this,&ofApp::guiConsoleEvent);

        ofSetLoggerChannel(ofPtr<ofGUILoggerChannel>(new ofGUILoggerChannel(this)));


        // scripts to run
        scripts.push_back("scripts/dragScript.lua");

        reset_directory_gui();

        currentScript = 0;

        // init the lua state
        lua.init(true);

        // listen to error events
        lua.addListener(this);

        ofGetFrameNum();

        // bind the OF api to the lua state
        lua.bind<ofxLuaBindings>();

        // run a script
        lua.doScript(scripts[currentScript]);

        // call the script's setup() function
        lua.scriptSetup();

        if(ofIsGLProgrammableRenderer()){
            ofLog()<<"YEA I'm Programmable!"<<endl;
        }
    }

    //--------------------------------------------------------------
    void ofApp::update() {
        // call the script's update() function
        lua.scriptUpdate();
    }

    //--------------------------------------------------------------
```



```
    void ofApp::draw() {
        // call the script's draw() function
        lua.scriptDraw();

        if(hasError){
            ofDrawBitmapStringHighlight(error, 9, 9);
        }
    }

    //--------------------------------------------------------------
    void ofApp::exit() {
        // call the script's exit() function
        lua.scriptExit();

        // clear the lua state
        lua.clear();

        delete gui;
        delete guiConsole;
    }

    //--------------------------------------------------------------
    void ofApp::keyPressed(int key) {

        if ( key == OF_KEY_ESC ){
            if ( gui->isVisible() ){
                gui->toggleVisible();
                gui->clearWidgets();
                guiConsole->toggleVisible();
            }else{
                reset_directory_gui();
                guiConsole->toggleVisible();
            }
        }

        lua.scriptKeyPressed(key);
    }

    //--------------------------------------------------------------
    void ofApp::mouseMoved(int x, int y) {
        lua.scriptMouseMoved(x, y);
    }

    //--------------------------------------------------------------
    void ofApp::mouseDragged(int x, int y, int button) {
```

```
        lua.scriptMouseDragged(x, y, button);
    }

    //--------------------------------------------------------------
    void ofApp::mousePressed(int x, int y, int button) {
        lua.scriptMousePressed(x, y, button);
    }

    //--------------------------------------------------------------
    void ofApp::mouseReleased(int x, int y, int button) {
        lua.scriptMouseReleased(x, y, button);
    }

    //--------------------------------------------------------------
    // ofxLua error callback
    void ofApp::errorReceived(string& msg) {
        ofLogNotice() << "got a script error: " << msg;
        hasError = true;
        error = msg;

        addConsoleMessage(msg);
    }

    //--------------------------------------------------------------
    void ofApp::reloadScript() {
        // exit, reinit the lua state, and reload the current script
        hasError = false;
        lua.scriptExit();
        lua.init(true);
        lua.bind<ofxLuaBindings>(); // rebind

        //Clear the gui console
        guiConsole->clearWidgets();

        //add the current script path to the lua path so require works correctly
        string fullpath =
    ofFilePath::getAbsolutePath(ofToDataPath(scripts[currentScript]));
        string folder = ofFilePath::getEnclosingDirectory(fullpath);
        string new_path("package.path = '");
        new_path.append(folder);
        new_path.append("?.lua;' .. package.path;");
        lua.doString(new_path);

        ofResetElapsedTimeCounter();
        lua.doScript(scripts[currentScript]);
```



```
        lua.scriptSetup();
    }

    void ofApp::add_to_gui(string path){
        ofDirectory dir(path);
        if (!dir.isDirectory())
            return;

        //list all lua files, add gui for these
        dir.allowExt("lua");
        dir.listDir();
        for(int i = 0; i < dir.size(); ++i){
            string lua_file = dir.getPath(i);
            directory_map.insert(pair<string,string>(lua_file, path));
            gui->addButton(lua_file, false);
        }

        //list all directories and recursively appl this func
        dir = ofDirectory(path);
        dir.listDir();
        for(int i = 0; i < dir.size(); ++i){
            add_to_gui(dir.getPath(i));
        }
    }

    void ofApp::build_directory_gui(){
        string path = "./scripts/";
        gui->addLabel("./scripts/");
        add_to_gui(path);
    }

    void ofApp::reset_directory_gui(){
        build_directory_gui();
        gui->setVisible(true);
        gui->autoSizeToFitWidgets();
    }

    void ofApp::guiEvent(ofxUIEventArgs &e){
        string name = e.widget->getName();
        int kind = e.widget->getKind();
        if ( kind == OFX_UI_WIDGET_BUTTON && e.getButton()->getValue() == 0){
            scripts.clear();
            scripts.push_back(name);
            reloadScript();
        }
```

```
}

void ofApp::guiConsoleEvent(ofxUIEventArgs &e){

}

void ofApp::addConsoleMessage(const string& message){
    //guiConsole
    guiConsole->addLabel(message);
}

//--------------------------------------------------
void ofGUILoggerChannel::log(ofLogLevel level, const string & module, const string
& message){
    // print to cerr for OF_LOG_ERROR and OF_LOG_FATAL_ERROR, everything else to
       cout
    ostream& out = level < OF_LOG_ERROR ? cout : cerr;
    out << "[" << ofGetLogLevelName(level, true)  << "] ";
    // only print the module name if it's not ""
    if(module != ""){
        //out << module << ": ";
        return ofLog(level)<<module<<": "<<message;
    }else{
        app->addConsoleMessage(message);
    }
}

void ofGUILoggerChannel::log(ofLogLevel level, const string & module, const char*
format, ...){
    //TODO: this isn't supported yet by the gui console
    va_list args;
    va_start(args, format);
    log(level, module, format, args);
    va_end(args);
}

void ofGUILoggerChannel::log(ofLogLevel level, const string & module, const char*
format, va_list args){
    //thanks stefan!
    //http://www.ozzu.com/cpp-tutorials/tutorial-writing-custom-printf-wrapper-funct
ion-t89166.html
    FILE* out = level < OF_LOG_ERROR ? stdout : stderr;
    fprintf(out, "[%s] ", ofGetLogLevelName(level, true).c_str());
    if(module != ""){
        fprintf(out, "%s: ", module.c_str());
```



```
        }
        vfprintf(out, format, args);
        fprintf(out, "\n");
        //TODO: this isn't supported yet
    }
```

```cpp
//
//  ReverseString_unittest.cpp
//  CppTests
//
//  Created by Stewart Bracken on 12/7/13.
//  Copyright (c) 2013 Stewart Bracken. All rights reserved.
//

#include <stdio.h>
#include <gtest/gtest.h>
#include <string>
using namespace std;

//s is a null terminated c-style string
void reverseCStr (char* s){
    char* end = s;
    char* begin = s;
    while( * (end+1) != '\0') end++;
    while (begin != end){
        char tmp = *end;
        *end = *begin;
        *begin = tmp;
        begin++;
        if(begin == end) break;
        end--;
    }
    return;
}

TEST(ReverseCString_unittest, ReverseCString){
    char s[4] = {'a','b','c','\0'};
    std::cout << "Orig: "<< string(s) <<std::endl;
    reverseCStr(s);
    std::cout<< "Result: " << string(s)<<std::endl;
    string str(s);
}
```

```java
package com.blindtigergames.werescrewed.entity.mover;

import com.badlogic.gdx.math.Vector2;
import com.badlogic.gdx.physics.box2d.Body;
import com.blindtigergames.werescrewed.entity.Entity;
import com.blindtigergames.werescrewed.entity.platforms.Platform;
import com.blindtigergames.werescrewed.entity.screws.Screw;

public class DirectionFlipMover implements IMover {

    boolean moveLeft;
    Vector2 impulse;
    float prevXPosMeter;
    float accum, timeToFlipAfterNoMove, maxSpeed;

    //private variables to prevent re-allocating them each time move() is called
    private float pos, diff, len;

    /**
     * Attach this mover to a dynamic body. IT will roll it left and right, and
flip directions if stuck on a wall
     * @param moveLeft Starting direction
     * @param impulseStrength 0.001f is a good slow acceleration speed
     * @param entityToMove Must be dynamic
     * @param timeToFlipAfterNoMove seconds to flip after being stuck on a wall.
1.5 is a good time.
     * @param maxSpeed 0.03 is a good speed
     */
    public DirectionFlipMover(boolean moveLeft, float impulseStrength, Entity
entityToMove, float timeToFlipAfterNoMove, float maxSpeed){
        this.moveLeft=moveLeft;
        this.impulse=new Vector2(impulseStrength,0);
        if(moveLeft)impulse.x*=-1;
        this.prevXPosMeter = entityToMove.getPosition( ).x;
        this.accum = 0;
        this.timeToFlipAfterNoMove=timeToFlipAfterNoMove;
        this.maxSpeed=maxSpeed;
    }

    /**
     * Initialize this mover with default values
     * @param moveLeft
     * @param entityToMove
     */
    public DirectionFlipMover(boolean moveLeft, Entity entityToMove){
```



```java
        this(moveLeft, 0.001f, entityToMove, 1.5f, .03f);
    }


    @Override
    public void move( float deltaTime, Body body ) {
        pos = body.getPosition( ).x;
        diff = pos- prevXPosMeter ;
        len = Math.abs( diff );
        if (len< 0.01f){ //0.01 means the enemy hasn't move much
            accum+=deltaTime;
        }
        prevXPosMeter=pos;
        if(accum>timeToFlipAfterNoMove){
            moveLeft = !moveLeft;
            accum = 0;
            impulse.x=impulse.x*-1;
        }
        if(len<maxSpeed){
            body.applyLinearImpulse( impulse, body.getWorldCenter( ) );
        }
    }

    @Override
    public void runPuzzleMovement( Screw screw, float screwVal, Platform p ) {
        // TODO Auto-generated method stub

    }

    @Override
    public PuzzleType getMoverType( ) {
        // TODO Auto-generated method stub
        return PuzzleType.OVERRIDE_ENTITY_MOVER;
    }

}
```

```
package com.blindtigergames.werescrewed.entity.mover;

import com.badlogic.gdx.math.Vector2;
import com.badlogic.gdx.physics.box2d.Body;
import com.blindtigergames.werescrewed.WereScrewedGame;
import com.blindtigergames.werescrewed.entity.Entity;
import com.blindtigergames.werescrewed.entity.Skeleton;
import com.blindtigergames.werescrewed.entity.platforms.Platform;
import com.blindtigergames.werescrewed.entity.screws.Screw;
import com.blindtigergames.werescrewed.sound.SoundManager;
import com.blindtigergames.werescrewed.sound.SoundManager.SoundRef;
import com.blindtigergames.werescrewed.util.Util;

public class CannonLaunchMover implements IMover {

    Skeleton cannon;
    float impulseStrength;
    float delay;
    SoundManager sounds;

    public CannonLaunchMover( Skeleton cannon, float impulseStrength,
            float delaySeconds ) {
        this.impulseStrength = impulseStrength;
        this.cannon = cannon;
        this.delay = delaySeconds;
        loadSounds();
    }

    @Override
    public void move( float deltaTime, Body body ) {
        delay -= deltaTime;
        if ( delay <= 0f ) {
            // Gdx.app.log( "CannonLaunchMover", "LAUNCHING!" );
            Vector2 impulseDirection = Util.PointOnCircle( impulseStrength,
                    cannon.body.getAngle( ) + Util.HALF_PI, new Vector2( ) );
            body.applyLinearImpulse( impulseDirection, body.getWorldCenter( ) );
            ( ( Entity ) ( body.getUserData( ) ) ).setMoverNullAtCurrentState( );
                // delete

    // this

    // mover!
            SoundRef launch = sounds.getSound( "launch" );
            launch.setVolume(1.0f);
            launch.play( true );
```



```
                cannon.addBehindParticleEffect( "cannon", true, true ).start( );
            }
            sounds.update(deltaTime);
        }

        @Override
        public void runPuzzleMovement( Screw screw, float screwVal, Platform p ) {
            // TODO Auto-generated method stub

        }

        @Override
        public PuzzleType getMoverType( ) {
            // TODO Auto-generated method stub
            return null;
        }

        public void loadSounds(){
            sounds = new SoundManager();
            sounds.getSound( "launch" , WereScrewedGame.dirHandle +
    "/levels/dragon/sounds/cannon.ogg");

        }
    }
```

```java
package com.blindtigergames.werescrewed.entity.builders;

import java.util.HashMap;

import com.badlogic.gdx.graphics.Texture;
//... omitted
import com.blindtigergames.werescrewed.util.ArrayHash;

/**
 * EntityBuilder is meant to simplify creating entities and allow for extension
 * through inheritance and polymorphism. Will probably be a constant
 * work-in-progress as new Entity classes are added.
 *
 * I added this generic version of EntityBuilder to better allow for different
 * types of builders. Now new subclasses of EntityBuilder don't have to redefine
 * its parent's methods; you just have to specify the new type in the "extends"
 * tag, and the generic will handle the rest for you.
 *
 * @author Kevin
 *
 */
public class GenericEntityBuilder< B extends GenericEntityBuilder< ? >> {

    // Common to all builders
    protected String name;
    protected Vector2 pos; // in pixels
    protected float rot;
    protected Vector2 sca;
    protected IMover mover;
    protected boolean solid;
    protected String definition;
    protected ArrayHash< String, HashMap< String, String >> sounds;
    protected Array<String> soundlines;

    // Used for type+world construction
    protected EntityDef type;
    protected World world;

    // Used for texture+body construction
    protected Texture tex;
    protected Body body;

    public GenericEntityBuilder( ) {
        resetInternal( );
    }
```



```java
    protected void resetInternal( ) {
        name = "";
        pos = new Vector2( 0, 0 );
        rot = 0.0f;
        sca = new Vector2( 1, 1 );
        solid = true;
        mover = null;
        type = null;
        world = null;
        tex = null;
        body = null;
        sounds = new ArrayHash< String, HashMap< String, String >>( );
        soundlines = new Array<String>();
        definition = "";
    }

    // Simply resets the builder to initial state and returns it.
    @SuppressWarnings( "unchecked" )
    public B reset( ) {
        resetInternal( );
        return ( B ) this;
    }

    /**
     *
     * @param name
     *            - String name of entity, default is "noname"
     * @return EntityBuilder
     */
    @SuppressWarnings( "unchecked" )
    public B name( String n ) {
        name = n;
        return ( B ) this;
    }

    /**
     *
     * @param definition
     *            - String XML name of entity, default is "noname"
     * @return EntityBuilder
     */
    @SuppressWarnings( "unchecked" )
    public B definition( String d ) {
        definition = d;
```

```
        return ( B ) this;
    }

    /**
     *
     * @param def
     *            - EntityDef used to load body/texture information.
     * @return EntityBuilder
     */
    @SuppressWarnings( "unchecked" )
    public B type( EntityDef def ) {
        type = def;
        if ( type.getCategory( ) == EntityCategory.PLAYER ) {
            return ( B ) new PlayerBuilder( ).copy( this );
        }
        return ( B ) this.properties( def.getProperties( ) );
    }

    /**
     *
     * @param def
     *            - Runs the EntityDef function with the definition loaded from
     *            this name.
     * @return EntityBuilder
     */
    public B type( String def ) {
        return ( B ) type( EntityDef.getDefinition( def ) );
    }

    /**
     *
     * @param world
     *            - sets the current world of the created entity.
     * @return EntityBuilder
     */
    @SuppressWarnings( "unchecked" )
    public B world( World w ) {
        world = w;
        return ( B ) this;
    }

    /**
     *
     * @param body
     *            - sets the body of the created entity.
```



```
     * @return EntityBuilder
     */
    @SuppressWarnings( "unchecked" )
    public B body( Body b ) {
        body = b;
        world = b.getWorld( );
        return ( B ) this;
    }

    /**
     *
     * @param tex
     *            - sets the texture of the created entity.
     * @return EntityBuilder
     */
    @SuppressWarnings( "unchecked" )
    public B texture( Texture t ) {
        tex = t;
        return ( B ) this;
    }

    /**
     *
     * @param p
     *            - sets the position of the created entity in PIXELS.
     * @return EntityBuilder
     */
    @SuppressWarnings( "unchecked" )
    public B position( Vector2 p ) {
        return ( B ) positionX( p.x ).positionY( p.y );
    }

    /**
     *
     * @param x
     *            - new x position of the created entity (in pixels)
     * @param y
     *            - new y position of the created entity (in pixels)
     * @return EntityBuilder
     */
    @SuppressWarnings( "unchecked" )
    public B position( float x, float y ) {
        return ( B ) positionX( x ).positionY( y );
    }
```

```
    /**
     *
     * @param x
     *            - new x position of the created entity in PIXELS.
     * @return EntityBuilder
     */
    @SuppressWarnings( "unchecked" )
    public B positionX( float x ) {
        pos.x = x;
        return ( B ) this;
    }

    /**
     *
     * @param y
     *            - new y position of the created entity in PIXELS.
     * @return EntityBuilder
     */
    @SuppressWarnings( "unchecked" )
    public B positionY( float y ) {
        pos.y = y;
        return ( B ) this;
    }

    /**
     *
     * @param r
     *            - new angle of the created entity in radians
     * @return EntityBuilder
     */
    @SuppressWarnings( "unchecked" )
    public B rotation( float r ) {
        rot = r;
        return ( B ) this;
    }

    /**
     *
     * @param s
     *            - sets whether the created entity is solid or not.
     * @return EntityBuilder
     */
    @SuppressWarnings( "unchecked" )
    public B solid( boolean s ) {
        solid = s;
```



```
        return ( B ) this;
    }

    /**
     * Loads an entity's special properties from a hashmap.
     *
     * @param props
     *            - String/String hashmap containing the data
     * @return EntityBuilder
     */
    @SuppressWarnings( "unchecked" )
    public B properties( ArrayHash< String, String > props ) {
        if ( props.containsKey( "texture" ) ) {
            this.texture( WereScrewedGame.manager.get( props.get( "texture" ),
                    Texture.class ) );
        }
        // Handle sound tags
        boolean moreSounds = true;
        String tag;
        for (int i = -1; i < 99 && moreSounds; i++){
            if (i < 0){
                tag = "sound";
            } else {
                tag = "sound" + i;
            }
            if (props.containsKey( tag )){
                for ( String line : props.getAll( tag ) ) {
                    soundlines.add( line );
                }
            } else if (i >= 0){
                moreSounds = false;
            }
        }
        return ( B ) this;
    }

    /**
     * Data-wise copy of another EntityBuilder into this one.
     *
     * @param that
     *            - the original builder to be copied.
     * @return EntityBuilder
     */
    @SuppressWarnings( "unchecked" )
    public B copy( GenericEntityBuilder< ? > that ) {
```

```
            name = that.name;
            pos = that.pos;
            rot = that.rot;
            sca = that.sca;
            solid = that.solid;
            mover = that.mover;
            type = that.type;
            world = that.world;
            tex = that.tex;
            body = that.body;
            return ( B ) this;
        }

        /**
         * Returns whether the builder has enough information to build. For most
         * entities, you need a world and either a Body or an EntityDef.
         *
         * @return boolean
         */
        protected boolean canBuild( ) {
            if ( world == null )
                return false;
            if ( type == null && body == null )
                return false;
            return true;
        }

        /**
         * Returns the reason (if any) the builder does not have enough information
         * to build. Returns empty string if no problems were found.
         *
         * @return String
         */
        protected String whyCantBuild( ) {
            if ( world == null )
                return "World is null.";
            if ( type == null && body == null )
                return "No type/body specified.";
            return "";
        }

        /**
         * Returns an entity created from given data.
         *
         * @return Entity
```



```
         */
        public Entity build( ) {
            Entity out = null;
            if ( canBuild( ) ) {
                if ( type != null ) {
                    out = new Entity( name, type, world, pos, rot, sca, tex, solid );
                } else {
                    out = new Entity( name, pos, tex, body, solid );
                }
            }
            prepareEntity( out );
            return out;
        }

        protected void prepareEntity( Entity out ) {
            if ( out != null ) {
                if ( mover != null ) {
                    out.addMover( mover, RobotState.IDLE );
                }
                if ( soundlines.size > 0 ) {
                    SoundManager soundMan = out.getSoundManager( );
                    if ( soundMan == null ) {
                        soundMan = new SoundManager( );
                        out.setSoundManager( soundMan );
                    }
                    for (String line: soundlines){
                        soundMan.getSoundWithProperties( line );
                    }
                }
                out.postLoad( );
            }
        }

        protected static final String nameTag = "Name";
        protected static final String typeTag = "Definition";
        protected static final String xTag = "X";
        protected static final String yTag = "Y";
        protected static final String aTag = "Angle";

    }
```

```java
package com.blindtigergames.werescrewed.entity.tween;

import aurelienribon.tweenengine.Timeline;
import aurelienribon.tweenengine.Tween;
import aurelienribon.tweenengine.TweenEquation;
import aurelienribon.tweenengine.TweenEquations;

import com.blindtigergames.werescrewed.entity.mover.TimelineTweenMover;
import com.blindtigergames.werescrewed.entity.platforms.Platform;

/**
 * Builds simple paths for platforms to move on. Use pixels for positions and
 * all positions are relative to the platform's spawning location. Support for
 * rotation depends on requests for it's use. Ask Stew if you really want
 * rotation support.
 *
 * @author stew
 *
 */
public class PathBuilder {
    private Timeline timeline;
    private Platform platformToMove;
    private TweenEquation easeFunction;
    private boolean repeatYoyo;
    private float timelineDelay;
    private float delay;
    private int loopCount;

    public PathBuilder( ) {
        reset( );
    }

    public void reset( ) {
        timeline = null;
        platformToMove = null;
        easeFunction = TweenEquations.easeNone;
        repeatYoyo = false;
        timelineDelay = 0f;
        delay = 0f;
        loopCount = Tween.INFINITY;
    }

    /**
     * start your path. you better use PathBuilder.platform() before you set a
     * target.
```



```java
     *
     * @return
     */
    public PathBuilder begin( ) {
        this.timeline = Timeline.createSequence( );
        return this;
    }

    /**
     * start your path using this and the path will apply to this platform
     *
     * @param platform
     *            platform to apply path to.
     * @return
     */
    public PathBuilder begin( Platform platform ) {
        this.platformToMove = platform;
        return this.begin( );
    }

    /**
     * set the target of the next target on the path
     *
     * @param platformToMove
     * @return
     */
    public PathBuilder platform( Platform platformToMove ) {
        this.platformToMove = platformToMove;
        return this;
    }

    /**
     * Set the ease of all subsequent targets on this path.
     *
     * @param easeFunction
     * @return
     */
    public PathBuilder ease( TweenEquation easeFunction ) {
        this.easeFunction = easeFunction;
        return this;
    }

    /**
     * set a new target on the path for the platform. Happens after the target
     * before and before the target after.
```

```
     *
     * @param xPixel
     *            PIXELS!!
     * @param yPixel
     *            PIXELS!!
     * @param time
     *            time to reach target from prev target (speed)
     * @return
     */
    public PathBuilder target( float xPixel, float yPixel, float time ) {
        timeline.push( Tween
                .to( platformToMove, PlatformAccessor.LOCAL_POS_XY, time )
                .delay( delay ).target( xPixel, yPixel )
                .ease( this.easeFunction ).start( ) );
        return this;
    }

    public PathBuilder repeatYoyo( boolean wantYoyoRepeat ) {
        this.repeatYoyo = wantYoyoRepeat;
        return this;
    }

    /**
     * the delay for each waypoint on the timeline 0 by default. applies to
     * every waypoint afterwards unless set back to 0.
     *
     * @param pathDelay
     * @return
     */
    public PathBuilder delay( float pathDelay ) {
        delay = pathDelay;
        return this;
    }

    /**
     * After each timeline loops, this delay will follow. 0 by default
     *
     * @param timelineDelay
     * @return
     */
    public PathBuilder timelineDelay( float timelineDelay ) {
        this.timelineDelay = timelineDelay;
        return this;
    }
```



```
    /**
     * set the number of loops of this timeline. infinity by default.
     *
     * @param loopCount
     * @return
     */
    public PathBuilder loops( int loopCount ) {
        this.loopCount = loopCount;
        return this;
    }

    /**
     * builds and returns the path you created. Pass this into a timeline mover.
     *
     * @return
     */
    public TimelineTweenMover build( ) {
        if ( repeatYoyo ) {
            timeline = timeline.repeatYoyo( loopCount, timelineDelay );
        } else {
            timeline = timeline.repeat( loopCount, timelineDelay );
        }

        return new TimelineTweenMover( timeline.start( ) );
    }

}
```

```java
package com.blindtigergames.werescrewed.entity.platforms;

import com.badlogic.gdx.graphics.Texture;
// omitted
import com.blindtigergames.werescrewed.util.Util;

/**
 * Platform Mostly just an inherited class, but complex platform uses that as
 * it's main class
 *
 * @author Ranveer / Stew
 *
 */

public class Platform extends Entity {

    // ============================================
    // Fields
    // ============================================
    protected float width, height;
    protected boolean dynamicType = false;
    protected boolean rotate = false;
    public boolean oneSided = false;
    public boolean moveable = false;
    // tileConstant is 16 for setasbox function which uses half width/height
    // creates 32x32 objects
    protected static final int tileConstant = 16;
    /**
     * Use this for any tile size calculations
     */
    public static final int tile = 32;

    protected PlatformType platType;

    /**
     * Used for kinematic movement connected to skeleton. Pixels.
     */
    protected Vector2 localPosition; // in pixels, local coordinate system
    protected Vector2 previousPosition;
    protected Vector2 prevBodyPos;
    private float localRotation; // in radians, local rot system
    protected float previousRotation;
    protected float prevBodyAngle;
    protected Vector2 localLinearVelocity; // in meters/step
    protected float localAngularVelocity; //
```



```java
    protected Vector2 originPosition; // world position that this platform
                                      // spawns
                                      // at, in pixels

    private Vector2 originRelativeToSkeleton; // box meters

    protected Joint extraSkeletonJoint;
    private boolean firstStep = true;

    // ============================================
    // Constructors
    // ============================================

    /**
     * General purpose platform constructor for things that don't use an
     * entitydef. Currently used by PlatformBuilder and Tiled Platform
     *
     * @param name
     * @param pos
     * @param tex
     * @param world
     */
    public Platform( String name, Vector2 pos, Texture tex, World world ) {
        super( name, pos, tex, null, true );
        this.world = world;
        entityType = EntityType.PLATFORM;
        init( pos );
    }

    /**
     * Construct platforms using an EntityDef. This is used by
     * PlatformBuilder.buildComplexBody()
     *
     * @param name
     * @param type
     * @param world
     * @param pos
     * @param rot
     * @param scale
     */
    public Platform( String name, EntityDef type, World world, Vector2 pos,
            float rot, Vector2 scale ) {
        super( name, type, world, pos, rot, scale, null, true );
        entityType = EntityType.PLATFORM;
        init( pos );
```

```
    }

    /**
     * Loading a Complex platform, or used to load complex Hazard
     *
     * (no scale or rotation because its defined in entitydef)
     *
     * @param name
     * @param type
     * @param world
     * @param pos
     */

    public Platform( String name, EntityDef type, World world, Vector2 pos ) {
        super( name, type, world, pos, null );
        entityType = EntityType.PLATFORM;
        init( pos );
    }

    /**
     * Initialize things.
     *
     * @author stew
     * @param pos
     */
    void init( Vector2 pos ) {
        localPosition = new Vector2( 0, 0 );
        previousPosition = new Vector2( localPosition.x, localPosition.y );
        prevBodyPos = new Vector2( 0, 0 );
        localLinearVelocity = new Vector2( 0, 0 );
        localRotation = 0;
        previousRotation = localRotation;
        originPosition = pos.cpy( );
        platType = PlatformType.DEFAULT; // set to default unless subclass sets
                                         // it later in a constructor
        originRelativeToSkeleton = new Vector2( );
    }

    // ===========================================
    // Methods
    // ===========================================

    /**
     * return localPosition Vector2 in PIXELS.
     *
```



```
     * @return
     */
    public Vector2 getLocalPos( ) {
        return localPosition;
    }

    /**
     * set localPosition Vector2 in PIXELS!!!
     *
     * @param newLocalPos
     *            in PIXELS
     */
    public void setLocalPos( Vector2 newLocalPosPixel ) {
        setLocalPos( newLocalPosPixel.x, newLocalPosPixel.y );
    }

    public void setLocalPos( float xPixel, float yPixel ) {
        localPosition.x = xPixel;
        localPosition.y = yPixel;
    }

    /**
     * returns local rotation in RADIANS
     */
    public float getLocalRot( ) {
        return localRotation;
    }

    /**
     * returns previous location last time it moved
     */
    public boolean hasMoved( ) {
        Vector2 bodyPos = body.getPosition( ).mul( Util.BOX_TO_PIXEL );
        if ( previousPosition.x != localPosition.x
                || previousPosition.y != localPosition.y
                || ( body != null && ( prevBodyPos.x != bodyPos.x || prevBodyPos.y
!= bodyPos.y ) ) ) {
            return true;
        }
        return false;
    }

    /**
     * set the previous position to this position
     */
```

```
    public void setPreviousTransformation( ) {
        Vector2 bodyPos = body.getPosition( ).mul( Util.BOX_TO_PIXEL );
        previousPosition = new Vector2( localPosition.x, localPosition.y );
        if ( body != null ) {
            prevBodyPos = new Vector2( bodyPos.x, bodyPos.y );
            prevBodyAngle = body.getAngle( );
        }
        previousRotation = localRotation;
    }

    /**
     * returns previous rotation last time it rotated
     */
    public boolean hasRotated( ) {
        if ( previousRotation != localRotation
                || prevBodyAngle != body.getAngle( ) ) {
            return true;
        }
        return false;
    }

    @Override
    public void updateDecals( float deltaTime ) {
        if ( firstStep || hasMoved( ) || hasRotated( ) || this.currentMover( ) !=
null ||
                ( this.getParentSkeleton( ) != null && ( this.getParentSkeleton(
).hasMoved( ) ||
                this.getParentSkeleton( ).hasRotated( )
                || this.getParentSkeleton( ).currentMover( ) != null ) ) ) {
            Vector2 bodyPos = this.getPositionPixel( );
            float angle = this.getAngle( ), cos = ( float ) Math.cos( angle ), sin
= ( float ) Math
                    .sin( angle );
            float x, y, r;
            Vector2 offset;
            Sprite decal;
            float a = angle * Util.RAD_TO_DEG;
            for ( int i = 0; i < fgDecals.size( ); i++ ) {
                offset = fgDecalOffsets.get( i );
                decal = fgDecals.get( i );
                r = fgDecalAngles.get( i );
                x = bodyPos.x + ( ( offset.x ) * cos ) - ( ( offset.y ) * sin );
                y = bodyPos.y + ( ( offset.y ) * cos ) + ( ( offset.x ) * sin );
                decal.setPosition( x + decal.getOriginX( ),
                        y + decal.getOriginY( ) );
```



```
                decal.setRotation( r + a );
            }
            for ( int i = 0; i < bgDecals.size( ); i++ ) {
                offset = bgDecalOffsets.get( i );
                decal = bgDecals.get( i );
                r = bgDecalAngles.get( i );
                x = bodyPos.x + ( ( offset.x ) * cos ) - ( ( offset.y ) * sin );
                y = bodyPos.y + ( ( offset.y ) * cos ) + ( ( offset.x ) * sin );
                decal.setPosition( x + decal.getOriginX( ),
                        y + decal.getOriginY( ) );
                decal.setRotation( r + a );
            }
        }
        firstStep = false;
    }

    /**
     * set local rotation in RADIAN
     *
     * @param newLocalRotRadians
     */
    public void setLocalRot( float newLocalRotRadians ) {
        localRotation = newLocalRotRadians;
    }

    /**
     * return originPosition Vector2 in PIXELS.
     *
     * @return
     */
    public Vector2 getOriginPos( ) {
        return originPosition;
    }

    /**
     * set Origin Position Vector2 in PIXELS!!!
     *
     * @param newLocalPos
     *            in PIXELS
     */
    public void setOriginPos( Vector2 newOriginPosPixel ) {
        originPosition.x = newOriginPosPixel.x;
        originPosition.y = newOriginPosPixel.y;
    }
```

```
    public void setOriginPos( float xPixel, float yPixel ) {
        originPosition.x = xPixel;
        originPosition.y = yPixel;
    }

    public Vector2 getLocLinearVel( ) {
        return localLinearVelocity;
    }

    public void setLocLinearVel( Vector2 linVelMeters ) {
        localLinearVelocity = linVelMeters.cpy( );
    }

    public void setLocLinearVel( float xMeter, float yMeter ) {
        localLinearVelocity.x = xMeter;
        localLinearVelocity.y = yMeter;
    }

    public float getLocAngularVel( ) {
        return localAngularVelocity;
    }

    public void setLocAngularVel( float angVelMeter ) {
        localAngularVelocity = angVelMeter;
    }

    @Override
    public void setAwake( ) {
        body.setAwake( true );
    }

    @Override
    public void update( float deltaTime ) {
        super.update( deltaTime );
        if ( removeNextStep ) {
            remove( );
        }
    }

    /**
     * Swap from kinematic to dynamic.
     */
    public void changeType( ) {
        dynamicType = !dynamicType;
        if ( dynamicType ) {
```



```
            body.setType( BodyType.DynamicBody );
            // Filter filter = new Filter( );
            // for ( Fixture f : body.getFixtureList( ) ) {
            // filter = f.getFilterData( );
            // // move player back to original category
            // filter.categoryBits = Util.CATEGORY_PLATFORMS;
            // // player now collides with everything
            // filter.maskBits = Util.CATEGORY_EVERYTHING;
            // f.setFilterData( filter );
            // }
        } else {
            body.setType( BodyType.KinematicBody );
            // Filter filter = new Filter( );
            // for ( Fixture f : body.getFixtureList( ) ) {
            // filter = f.getFilterData( );
            // // move player back to original category
            // filter.categoryBits = Util.CATEGORY_PLATFORMS;
            // // player now collides with everything
            // filter.maskBits = Util.CATEGORY_EVERYTHING;
            // f.setFilterData( filter );
            // }
        }

        body.setActive( false );
    }

    // This function sets the platform to 180* no matter what angle it currently
    // is
    public void setHorizontal( ) {
        body.setTransform( body.getPosition( ), ( float ) Math.toRadians( 90 ) );
    }

    // This function sets platform to 90*
    public void setVertical( ) {
        body.setTransform( body.getPosition( ), ( float ) Math.toRadians( 180 ) );
    }

    public boolean getOneSided( ) {
        return oneSided;
    }

    public void setOneSided( boolean value ) {
        oneSided = value;
    }
```

```
    protected void rotate( ) {
        body.setAngularVelocity( 1f );
    }

    protected void rotateBy90( ) {
        float bodyAngle = body.getAngle( );
        body.setTransform( body.getPosition( ), bodyAngle + 90 );
    }

    /**
     * Returns the private member platform type for casting or whatever
     *
     * @return PLATFORMTYPE
     */
    public PlatformType getPlatformType( ) {
        return platType;
    }

    /**
     * Set this platforms type!!
     *
     * @author stew
     * @param newPlatformType
     */
    public void setPlatformType( PlatformType newPlatformType ) {
        platType = newPlatformType;
    }

    /**
     * Set the position and angle of the kinematic platform based on the parent
     * skeleton's pos/rot. Now better than ever! Use this to set a platform's
     * velocity so the platform does normal phsyics.
     *
     * @param frameRate
     *            which is typically 1/deltaTime.
     * @param skeleton
     *
     * @author stew
     */
    public void setTargetPosRotFromSkeleton( float frameRate, Skeleton skeleton ) {
        if ( skeleton != null ) {
            Vector2 posOnSkeleLocalMeter = originRelativeToSkeleton.cpy( ).add(
                    localPosition.cpy( ).mul( Util.PIXEL_TO_BOX ) );
            float radiusFromSkeletonMeters = posOnSkeleLocalMeter.len( );
            float newAngleFromSkeleton = skeleton.body.getAngle( )
```



```
                    + Util.angleBetweenPoints( Vector2.Zero,
                            posOnSkeleLocalMeter );

            Vector2 targetPosition = Util.PointOnCircle(
                    radiusFromSkeletonMeters, newAngleFromSkeleton,
                    skeleton.getPosition( ) ).sub( body.getPosition( ) );
            float targetRotation = localRotation + skeleton.body.getAngle( )
                    - body.getAngle( );

            body.setLinearVelocity( targetPosition.mul( frameRate ) );
            body.setAngularVelocity( targetRotation * frameRate );
        }
    }

    /**
     * This function TRANSLATES a platform, so it won't act with normal physics.
     * This is mainly used for event triggers.
     *
     * @param skeleton
     * @author stew
     */
    public void translatePosRotFromSKeleton( Skeleton skeleton ) {
        if ( skeleton != null ) {
            Vector2 posOnSkeleLocalMeter = originRelativeToSkeleton.cpy( ).add(
                    localPosition.cpy( ).mul( Util.PIXEL_TO_BOX ) );

            if ( posOnSkeleLocalMeter.equals( Vector2.Zero ) ) {
                body.setTransform( skeleton.body.getPosition( ), localRotation
                        + skeleton.body.getAngle( ) );
            } else {
                float radiusFromSkeletonMeters = posOnSkeleLocalMeter.len( );
                float newAngleFromSkeleton = skeleton.body.getAngle( );
                newAngleFromSkeleton += Util.angleBetweenPoints( Vector2.Zero,
                        posOnSkeleLocalMeter );

                Vector2 targetPosition = Util.PointOnCircle(
                        radiusFromSkeletonMeters, newAngleFromSkeleton,
                        skeleton.getPosition( ) );
                float targetRotation = localRotation + skeleton.body.getAngle( );

                body.setTransform( targetPosition, targetRotation );
            }

        }
    }
```

```
    @Override
    public void setCrushing( boolean value ) {
        crushing = value;
        oneSided = false;
    }

    public Vector2 getOriginRelativeToSkeleton( ) {
        return originRelativeToSkeleton;
    }

    public void setOriginRelativeToSkeleton( Vector2 originRelativeToSkeleton ) {
        this.originRelativeToSkeleton = originRelativeToSkeleton;
    }

    public void constructBodyFromVerts( Array< Vector2 > loadedVerts,
            Vector2 positionPixel ) {
        BodyDef bodyDef = new BodyDef( );
        bodyDef.position.set( positionPixel.mul( Util.PIXEL_TO_BOX ) );
        body = world.createBody( bodyDef );

        PolygonShape polygon = new PolygonShape( );
        Vector2[ ] verts = new Vector2[ loadedVerts.size - 1 ];

        // MAKE SURE START POINT IS IN THE MIDDLE
        // AND SECOND AND END POINT ARE THE SAME POSITION
        int i = 0;
        for ( int j = 0; j < loadedVerts.size; j++ ) {
            if ( j == loadedVerts.size - 1 )
                continue;
            Vector2 v = loadedVerts.get( j );
            verts[ i ] = new Vector2( v.x * Util.PIXEL_TO_BOX, v.y
                    * Util.PIXEL_TO_BOX );
            ++i;
        }
        polygon.set( verts );

        FixtureDef fixture = new FixtureDef( );
        fixture.shape = polygon;

        body.createFixture( fixture );
        body.setUserData( this );

        polygon.dispose( );
    }
```



```
    /**
     * This function is used to joint a platform to a skeleton so that it stays
     * in place also this way we save the reference to that particular joint so
     * we can delete it later
     *
     * @param skel
     */
    public void addJointToSkeleton( Skeleton skel ) {
        RevoluteJointDef rjd = new RevoluteJointDef( );
        rjd.initialize( body, skel.body, body.getWorldCenter( ) );
        extraSkeletonJoint = ( Joint ) this.world.createJoint( rjd );
    }

    /**
     * Adds the joint (connected to a skeleton) to the list to remove it when
     * the Box2d world is not locked() (otherwise it crashes)
     *
     * Only really used when level loading
     */
    public void destorySkeletonJoint( ) {
        if ( extraSkeletonJoint != null ) {
            Level.jointsToRemove.add( extraSkeletonJoint );
            extraSkeletonJoint = null;
        }
    }

}
```

```
/**
 *
 */
package com.blindtigergames.werescrewed.entity;

import static com.badlogic.gdx.graphics.g2d.SpriteBatch.X1;
import static com.badlogic.gdx.graphics.g2d.SpriteBatch.X2;
import static com.badlogic.gdx.graphics.g2d.SpriteBatch.X3;
import static com.badlogic.gdx.graphics.g2d.SpriteBatch.X4;
import static com.badlogic.gdx.graphics.g2d.SpriteBatch.Y1;
import static com.badlogic.gdx.graphics.g2d.SpriteBatch.Y2;
import static com.badlogic.gdx.graphics.g2d.SpriteBatch.Y3;
import static com.badlogic.gdx.graphics.g2d.SpriteBatch.Y4;

import com.badlogic.gdx.Gdx;
// [omitted]
import com.blindtigergames.werescrewed.graphics.SpriteBatch;

/**
 * A sprite that fills a texture inside a CONVEX polygon. Since this doesn't
 * support origin, you'll want to position your origin point at (0,0) and all
 * other points relative to that. This should behave just like a lib gdx sprite.
 *
 * @author Stew   a little bit Kevin :D
 *
 */
public class PolySprite extends Sprite {

    protected Mesh mesh;
    protected ShaderProgram shader;
    protected float[ ] verts;
    private Array< Vector2 > localVerts;
    private int numVerts;
    protected Rectangle bounds;
    protected Vector2 center;
    private float x, y;
    private float rotation;
    // private Matrix4 modelMat; //holds position and rotation for the polygon.
    // private Matrix4 rotationMat;

    private final int R = 3, G = 4, B = 5, A = 6, U = 7, V = 8, X = 0, Y = 1,
            Z = 2;

    /**
     * Construct a polysprite with a given texture and color
```



```
     *
     * @param texture
     *            , size/shape doesn't matter, it will be repeated
     * @param verts
     *            an array of verts, each vector2 is x,y of a vertice in pixels
     * @param r
     *            red color
     * @param g
     *            green
     * @param b
     *            blue
     * @param a
     *            alpha
     */
    public PolySprite( Texture texture, Array< Vector2 > verts, float r,
            float g, float b, float a ) {
        super( texture );
        init( verts );
        constructMesh( verts, r, g, b, a );
    }

    /**
     * Construct a polysprite with a texture and vertices given in pixels. The
     * color will be white.
     *
     * @param texture
     *            texture, size/shape doesn't matter, it will be repeated
     * @param verts
     *            an array of verts, each vector2 is x,y of a vertice in pixels
     */
    public PolySprite( Texture texture, Array< Vector2 > verts ) {
        super( texture );
        init( verts );
        constructMesh( verts, 1, 1, 1, 1 );
    }

    private void init( Array< Vector2 > verts ) {
        this.numVerts = verts.size;
        this.localVerts = verts;
        this.x = 0;
        this.y = 0;
        this.rotation = 0;
        // modelMat= new Matrix4( );
        // rotationMat = new Matrix4( ).rotate( 0, 0, 1, rotation );
    }
```

```
    @Override
    public void setPosition( float x, float y ) {
        translate( x - this.x, y - this.y );
    }

    /**
     * Sets the x position where the sprite will be drawn. If origin, rotation,
     * or scale are changed, it is slightly more efficient to set the position
     * after those operations. If both position and size are to be changed, it
     * is better to use {@link #setBounds(float, float, float, float)}.
     */
    @Override
    public void setX( float x ) {
        translateX( x - this.x );
    }

    /**
     * Sets the y position where the sprite will be drawn. If origin, rotation,
     * or scale are changed, it is slightly more efficient to set the position
     * after those operations. If both position and size are to be changed, it
     * is better to use {@link #setBounds(float, float, float, float)}.
     */
    @Override
    public void setY( float y ) {
        translateY( y - this.y );
    }

    /**
     * Sets the x position relative to the current position where the sprite
     * will be drawn. If origin, rotation, or scale are changed, it is slightly
     * more efficient to translate after those operations.
     */
    @Override
    public void translateX( float xAmount ) {
        // this.x += xAmount;
        translate( xAmount, 0 );
    }

    /**
     * Sets the y position relative to the current position where the sprite
     * will be drawn. If origin, rotation, or scale are changed, it is slightly
     * more efficient to translate after those operations.
     */
    @Override
```



```
    public void translateY( float yAmount ) {
        // y += yAmount;

        translate( 0, yAmount );

    }

    /**
     * Sets the position relative to the current position where the sprite will
     * be drawn.
     */
    @Override
    public void translate( float xAmount, float yAmount ) {
        x += xAmount;
        y += yAmount;

        float[ ] vertices = this.verts;

        for ( int i = 0; i < numVerts; i++ ) {
            vertices[ 9 * i ] += xAmount;
            vertices[ 9 * i + 1 ] += yAmount;
        }
        mesh.setVertices( verts );
    }

    /**
     * Set the color, form 0..1
     */
    @Override
    public void setColor( float r, float g, float b, float a ) {
        int intBits = ( ( int ) ( 255 * a ) << 24 )
                | ( ( int ) ( 255 * b ) << 16 ) | ( ( int ) ( 255 * g ) << 8 )
                | ( ( int ) ( 255 * r ) );
        float color = NumberUtils.intToFloatColor( intBits );
        float[ ] vertices = this.verts;

        for ( int i = 0; i < numVerts; i++ ) {
            vertices[ 9 * i + R ] = color; // r
            vertices[ 9 * i + G ] = color; // g
            vertices[ 9 * i + B ] = color; // b
            vertices[ 9 * i + A ] = color; // a
        }
    }

    @Override
```

```
    public void setColor( float color ) {
        float[ ] vertices = this.verts;
        for ( int i = 0; i < numVerts; i++ ) {
            vertices[ 9 * i + R ] = color; // r
            vertices[ 9 * i + G ] = color; // g
            vertices[ 9 * i + B ] = color; // b
            vertices[ 9 * i + A ] = color; // a
        }
    }

    /**
     * This could be a bug!
     */
    @Override
    public void setRotation( float degrees ) {
        rotate( degrees - rotation );
        // modelMat.idt( ).rotate( Vector3.Z, rotation ).translate( x, y, 0 );
    }

    /** Sets the sprite's rotation relative to the current rotation. */
    @Override
    public void rotate( float degrees ) {
        rotation += degrees;
        // modelMat.idt( ).rotate( Vector3.Z, rotation ).translate( x, y, 0 );

        float cos = MathUtils.cosDeg( rotation );
        float sin = MathUtils.sinDeg( rotation );

        for ( int i = 0; i < numVerts; i++ ) {
            float oldx = localVerts.get( i ).x;
            float oldy = localVerts.get( i ).y;
            verts[ 9 * i + X ] = ( oldx ) * cos - ( oldy ) * sin + this.x;
            verts[ 9 * i + Y ] = ( oldx ) * sin + ( oldy ) * cos + this.y;
        }
        mesh.setVertices( verts );
    }

    @Override
    public Rectangle getBoundingRectangle( ) {
        final float[ ] vertices = getVertices( );

        float minx = vertices[ X1 ];
        float miny = vertices[ Y1 ];
        float maxx = vertices[ X1 ];
        float maxy = vertices[ Y1 ];
```



```
        minx = minx > vertices[ X2 ] ? vertices[ X2 ] : minx;
        minx = minx > vertices[ X3 ] ? vertices[ X3 ] : minx;
        minx = minx > vertices[ X4 ] ? vertices[ X4 ] : minx;

        maxx = maxx < vertices[ X2 ] ? vertices[ X2 ] : maxx;
        maxx = maxx < vertices[ X3 ] ? vertices[ X3 ] : maxx;
        maxx = maxx < vertices[ X4 ] ? vertices[ X4 ] : maxx;

        miny = miny > vertices[ Y2 ] ? vertices[ Y2 ] : miny;
        miny = miny > vertices[ Y3 ] ? vertices[ Y3 ] : miny;
        miny = miny > vertices[ Y4 ] ? vertices[ Y4 ] : miny;

        maxy = maxy < vertices[ Y2 ] ? vertices[ Y2 ] : maxy;
        maxy = maxy < vertices[ Y3 ] ? vertices[ Y3 ] : maxy;
        maxy = maxy < vertices[ Y4 ] ? vertices[ Y4 ] : maxy;

        if ( bounds == null )
            bounds = new Rectangle( );
        bounds.x = minx;
        bounds.y = miny;
        bounds.width = maxx - minx;
        bounds.height = maxy - miny;

        // bounds.x = x-bounds.getWidth( )/2.0f;
        // bounds.y = y;

        return bounds;
    }

    @Override
    public void draw( SpriteBatch batch ) {
        // this should be called in render()
        if ( mesh == null )
            throw new IllegalStateException(
                    "drawMesh called before a mesh has been created." );

        GL20 gl = Gdx.graphics.getGL20( );
        if ( gl != null ) {
            // we don't necessarily need these, but its good practice to enable
            // the things we need. we enable 2d textures and set the active one
            // to 0. we could have multiple textures but we don't need it here.
            gl.glEnable( GL20.GL_TEXTURE_2D );
            gl.glActiveTexture( GL20.GL_TEXTURE0 );
```

```
            // setWrap also binds the texture.
            this.getTexture( ).setWrap( Texture.TextureWrap.Repeat,
                    Texture.TextureWrap.Repeat );
            // camera * modelview
            mesh.render( shader, GL20.GL_TRIANGLES );
        }
    }

    @Override
    public void setAlpha( float newAlpha ) {
        super.setAlpha( newAlpha );
        for ( int i = 0; i < numVerts; i++ ) {
            verts[ 9 * i + A ] = alpha;
        }
        mesh.setVertices( verts );
    }

    private void constructMesh( Array< Vector2 > verts, float r, float g,
            float b, float a ) {

        shader = WereScrewedGame.defaultShader;

        float minX = Float.MAX_VALUE;
        float minY = Float.MAX_VALUE;
        float maxX = Float.MIN_VALUE;
        float maxY = Float.MIN_VALUE;
        // 9 is 3 positions, 4 colors, and 2 texcoords
        this.verts = new float[ numVerts * 9 ];

        for ( int i = 0; i < numVerts; i++ ) {
            float x = verts.get( i ).x;
            float y = verts.get( i ).y;
            // get the bounds of the poly!
            if ( x < minX ) {
                minX = x;
            } else if ( x > maxX ) {
                maxX = x;
            }
            if ( y < minY ) {
                minY = y;
            } else if ( y > maxY ) {
                maxY = y;
            }
            this.verts[ 9 * i + X ] = x; // x
            this.verts[ 9 * i + Y ] = y; // y
```



```
            this.verts[ 9 * i + Z ] = 0; // z

            this.verts[ 9 * i + R ] = r; // r
            this.verts[ 9 * i + G ] = g; // g
            this.verts[ 9 * i + B ] = b; // b
            this.verts[ 9 * i + A ] = a; // a
        }

        this.bounds = new Rectangle( minX, minY, maxX - minX, maxY - minY );
        center = new Vector2( bounds.x + bounds.width / 2, bounds.y
                + bounds.height / 2 );

        float[ ] texCoords = createTexCoords( verts );

        for ( int i = 0; i < numVerts; i++ ) {
            this.verts[ 9 * i + U ] = texCoords[ 2 * i ]; // u
            this.verts[ 9 * i + V ] = texCoords[ 2 * i + 1 ]; // v
        }
        mesh = new Mesh( true, numVerts, ( numVerts - 2 ) * 3,
                VertexAttribute.Position( ), VertexAttribute.ColorUnpacked( ),
                VertexAttribute.TexCoords( 0 ) );
        mesh.setVertices( this.verts );
        mesh.setIndices( createIndices( ) );
    }

    /**
     * Creates a triangle fan array of indices for the given vertices
     *
     * @author stew
     * @param numVerts
     * @return
     */
    private short[ ] createIndices( ) {
        int numTriangles = numVerts - 2;
        // 3 indices per triangle, (numVerts-2) triangles
        short[ ] indices = new short[ numTriangles * 3 ];
        // insert the first triangle cus it's a shitty mc-weird initial case:
        indices[ 0 ] = 0;
        indices[ 1 ] = 1;
        indices[ 2 ] = 2;
        // then do the rest of the triangles:
        for ( short i = 1; i < numTriangles; ++i ) {
            indices[ i * 3 + X ] = ( short ) ( i + 1 );
            indices[ i * 3 + Y ] = ( short ) ( i + 2 );
            indices[ i * 3 + Z ] = 0;
```

```
            }
            return indices;
        }

        int getTextureWidth( ) {
            return getTexture( ).getWidth( );
        }

        int getTextureHeight( ) {
            return getTexture( ).getHeight( );
        }

        /**
         * really nicely lerp texture coordinates so the texture is not skewed on
         * the polygon.
         *
         * @param verts
         * @return
         */
        private float[ ] createTexCoords( Array< Vector2 > verts ) {
            float[ ] texCoords = new float[ verts.size * 2 ];
            float texWidth = bounds.width / getTextureWidth( );
            float texHeight = bounds.height / getTextureHeight( );
            float halfTexWidth = texWidth / 2;
            float halfTexHeight = texHeight / 2;

            for ( int i = 0; i < verts.size; ++i ) {
                texCoords[ 2 * i ] = verts.get( i ).x / bounds.width * texWidth
                        - halfTexWidth;
                texCoords[ 2 * i + 1 ] = verts.get( i ).y / bounds.height
                        * texHeight - halfTexHeight;
            }
            return texCoords;
        }
    }
```

```
package com.blindtigergames.werescrewed.entity;

import java.util.ArrayList;
import java.util.HashMap;

import com.badlogic.gdx.graphics.Texture;
// [omitted]
import com.blindtigergames.werescrewed.util.Util;

/**
 * A Skeleton is a node in the level tree structure. It moves platforms under it
 * as well as skeletons attached.
 *
 * @author Stewart
 *
 *         TODO: Perhaps change skeleton name, and make skeleton more like a
 *         tree (i.e. It should have a list of non-jointed entities too.)
 */

public class Skeleton extends Platform {

    // public static final int foreground = 0;
    // public static final int background = 1;
    // public static final int midground = 2;

    public PolySprite bgSprite, fgSprite;

    SimpleFrameAnimator alphaFadeAnimator;
    private final float fadeSpeed = 3f;

    protected HashMap< String, Platform > dynamicPlatformMap = new HashMap< String,
Platform >( );
    protected HashMap< String, Skeleton > childSkeletonMap = new HashMap< String,
Skeleton >( );
    protected HashMap< String, Platform > kinematicPlatformMap = new HashMap<
String, Platform >( );
    protected HashMap< String, Rope > ropeMap = new HashMap< String, Rope >( );
    protected HashMap< String, Screw > screwMap = new HashMap< String, Screw >( );
    protected HashMap< String, CheckPoint > checkpointMap = new HashMap< String,
CheckPoint >( );
    protected HashMap< String, EventTrigger > eventMap = new HashMap< String,
EventTrigger >( );
    private ArrayList< Entity > entitiesToRemove = new ArrayList< Entity >( );

    private int entityCount = 0;
```



```
    protected RootSkeleton rootSkeleton;
    protected Skeleton parentSkeleton;

    protected boolean applyFadeToFGDecals = true;
    protected boolean isMacroSkeleton = false;
    protected boolean invisibleBGDecal = false;

    protected boolean wasInactive = false;
    protected boolean onScreen = true;
    protected boolean isUpdatable = true;

    protected boolean setChildSkeletonsToSleep = false;
    protected boolean useBoundingRect = false;
    protected boolean updatedOnce = false;
    public Rectangle boundingRect = new Rectangle( -10000, -10000, 10000, 10000 );
    protected Rectangle lastCameraRect = new Rectangle( 0, 0, 0, 0 );
    protected boolean removed = false;

    public boolean respawningDontPutToSleep = false;

    private final float MAX_FALL_POS = -5000.0f;

    // private ShapeRenderer shapeRender;

    /**
     * Constructor used by SkeletonBuilder
     *
     * @param n
     * @param pos
     * @param tex
     * @param world
     * @param bodyType
     */
    public Skeleton( String n, Vector2 pos, Texture tex, World world,
            BodyType bodyType ) {
        super( n, pos, tex, world ); // not constructing body class
        this.world = world;
        constructSkeleton( pos, bodyType );
        super.setSolid( false );
        entityType = EntityType.SKELETON;
        alphaFadeAnimator = new SimpleFrameAnimator( ).speed( 0 )
                .loop( LoopBehavior.STOP ).time( 1 );
        // shapeRender = new ShapeRenderer( );
    }
```

```
    /**
     * COnstructor to default to kinematic body type
     *
     * @param n
     * @param pos
     * @param tex
     * @param world
     */
    public Skeleton( String n, Vector2 pos, Texture tex, World world ) {
        this( n, pos, tex, world, BodyType.KinematicBody );
    }

    public void constructSkeleton( Vector2 pos, BodyType bodyType ) {
        // Skeletons have no fixtures!!
        BodyDef skeletonBodyDef = new BodyDef( );
        skeletonBodyDef.type = bodyType;

        skeletonBodyDef.position.set( pos.cpy( ).mul( Util.PIXEL_TO_BOX ) );
        body = world.createBody( skeletonBodyDef );
        body.setUserData( this );

        FixtureDef dynFixtureDef = new FixtureDef( );
        PolygonShape polygon = new PolygonShape( );
        polygon.setAsBox( 100 * Util.PIXEL_TO_BOX, 100 * Util.PIXEL_TO_BOX );
        dynFixtureDef.shape = polygon;
        dynFixtureDef.density = 5f;
        dynFixtureDef.isSensor = true;
        dynFixtureDef.filter.categoryBits = Util.CATEGORY_SKELS;
        dynFixtureDef.filter.maskBits = Util.CATEGORY_SCREWS;
        body.createFixture( dynFixtureDef );
        polygon.dispose( );
        body.setGravityScale( 0.1f );
        // this.quickfixCollisions( );
    }

    /**
     * Attach a platform to this skeleton that will freely rotate about the
     * center. Make sure the platform is dynamic
     *
     * @param platform
     */
    public void addPlatformRotatingCenter( Platform platform ) {
        // Default values of the builder will allow rotation with anchor at
        // center of platform
```



```
        new RevoluteJointBuilder( world ).entityA( this ).entityB( platform )
                .build( );
        addDynamicPlatform( platform );
    }

    /**
     * Attach a platform to this skeleton that rotates with a motor the platform
     * must already be set as dynamic
     *
     * @param platform
     */
    public void addPlatformRotatingCenterWithMot( Platform platform,
            float rotSpeedInMeters ) {
        // Default values of the builder will allow rotation with anchor at
        // center of platform
        new RevoluteJointBuilder( world ).entityA( this ).entityB( platform )
                .motor( true ).motorSpeed( rotSpeedInMeters ).build( );

        addDynamicPlatform( platform );
    }

    /**
     * Add a platform that will only move / rotate with skeleton Don't use this.
     * if it's fixed, you might as well add it as kinematic
     *
     * @param platform
     */
    public void addDynamicPlatformFixed( Platform platform ) {
        new RevoluteJointBuilder( world ).entityA( this ).entityB( platform )
                .limit( true ).lower( 0 ).upper( 0 ).build( );
        addDynamicPlatform( platform );
    }

    /**
     * Add a platform to this skeleton. Will determine what list to add it to
     * for you!
     *
     * @param platform
     */
    public void addPlatform( Platform platform ) {
        if ( platform.body.getType( ) == BodyType.DynamicBody )
            addDynamicPlatform( platform );
        else
            addKinematicPlatform( platform );
    }
```

```
    public void addPlatforms( Platform... platforms ) {
        for ( Platform p : platforms ) {
            addPlatform( p );
        }
    }

    public void addRope( Rope rope, boolean toJoint ) {
        if ( toJoint ) {
            new RevoluteJointBuilder( world ).entityA( this )
                    .entityB( rope.getFirstLink( ) ).limit( true ).lower( 0 )
                    .upper( 0 ).build( );
        }
        // ropes.add( rope );
        ropeMap.put( rope.name, rope );
    }

    public boolean isMacroSkel( ) {
        return isMacroSkeleton;
    }

    public void setMacroSkel( boolean macroSkel ) {
        isMacroSkeleton = macroSkel;
    }

    /**
     *
     * @param ss
     *            - add stripped screw onto the skeleton
     */
    public void addStrippedScrew( StrippedScrew ss ) {
        addScrewForDraw( ss );
    }

    /**
     * Add a screw to be drawn!
     *
     * @param Screw
     */
    public void addScrewForDraw( Screw s ) {
        // screws.add(s);
        entityCount++;
        screwMap.put( s.name, s );
        s.setParentSkeleton( this );
    }
```



```
    /**
     * add checkpoint to be drawn
     */
    public void addCheckPoint( CheckPoint chkpt ) {
        entityCount++;
        checkpointMap.put( chkpt.name, chkpt );
        chkpt.setParentSkeleton( this );
    }

    /**
     * Simply adds a platform to the list, without explicitly attaching it to
     * the skelington
     *
     * @param Entity
     *            platform
     * @author stew
     */
    public void addDynamicPlatform( Platform platform ) {
        entityCount++;
        // this.dynamicPlatforms.add( platform );
        if ( dynamicPlatformMap.containsKey( platform.name ) ) {
            platform.name = getUniqueName( platform.name );
        }
        dynamicPlatformMap.put( platform.name, platform );
        platform.setParentSkeleton( this );
        platform.setOriginRelativeToSkeleton( platform.getPosition( ).cpy( )
                .sub( getPosition( ) ) );
    }

    /**
     * Add Kinamatic platform to this Skeleton
     *
     * @param Platform
     *            that's already set as kinematic
     */
    public void addKinematicPlatform( Platform platform ) {
        // kinematicPlatforms.add( platform );
        entityCount++;
        if ( kinematicPlatformMap.containsKey( platform.name ) ) {
            platform.name = getUniqueName( platform.name );
        }
        kinematicPlatformMap.put( platform.name, platform );
        platform.setParentSkeleton( this );
        platform.setOriginRelativeToSkeleton( platform.getPosition( ).cpy( )
```

```
                .sub( ( getPosition( ) ) ) );
    }

    public void addSteam( Steam steam ) {
        addKinematicPlatform( steam );
    }

    /**
     * Add EventTrigger to this Skeleton
     *
     * @param event
     *            EventTrigger to be added to Skeleton
     */
    public void addEventTrigger( EventTrigger event ) {
        entityCount++;
        if ( eventMap.containsKey( event.name ) ) {
            event.name = getUniqueName( event.name );
        }
        event.setParentSkeleton( this );
        event.setOriginRelativeToSkeleton( event.getPosition( ).cpy( )
                .sub( ( getPosition( ) ) ) );
        eventMap.put( event.name, event );
    }

    public void addHazard( Hazard h ) {
        addPlatform( h );
    }

    /**
     * Add a skeleton to the sub skeleton list of this one.
     *
     * @author stew
     */
    public void addSkeleton( Skeleton skeleton ) {
        // this.childSkeletons.add( skeleton );
        if ( this == rootSkeleton ) {
            skeleton.setMacroSkel( true );
        }
        skeleton.parentSkeleton = this;
        skeleton.rootSkeleton = this.rootSkeleton;
        childSkeletonMap.put( skeleton.name, skeleton );
        skeleton.setParentSkeleton( this );
        skeleton.setOriginRelativeToSkeleton( skeleton.getPosition( ).cpy( )
                .sub( ( getPosition( ) ) ) );
    }
```



```
    /**
     * set skeleton to awake or not TODO: Do kinamtic platforms need sleeping?
     */
    public void setSkeletonAwakeRec( boolean isAwake ) {
        for ( Skeleton skeleton : childSkeletonMap.values( ) ) {
            skeleton.setSkeletonAwakeRec( isAwake );
        }
        for ( Platform platform : dynamicPlatformMap.values( ) ) {
            platform.body.setAwake( isAwake );
        }
        for ( Platform platform : kinematicPlatformMap.values( ) ) {
            platform.body.setAwake( isAwake );
        }
        for ( Screw screw : screwMap.values( ) ) {
            screw.body.setAwake( isAwake );
        }
        for ( CheckPoint chkpt : checkpointMap.values( ) ) {
            chkpt.body.setAwake( isAwake );
        }
    }

    /**
     * finds the skeleton with this name
     */
    public Skeleton getSubSkeletonByName( String name ) {
        if ( childSkeletonMap.containsKey( name ) ) {
            return childSkeletonMap.get( name );
        }
        return null;
    }

    public void setSkeletonEntitiesToSleepRecursively( ) {
        this.setEntitiesToSleepOnUpdate( );
        this.wasInactive = true;
        for ( Skeleton skeleton : this.childSkeletonMap.values( ) ) {
            if ( !skeleton.dontPutToSleep ) {
                if ( this.useBoundingRect ) {
                    if ( inRectangleBounds( this.boundingRect,
                            skeleton.getPositionPixel( ) ) ) {
                        skeleton.setSkeletonEntitiesToSleepRecursively( );
                        skeleton.body.setActive( true );
                        skeleton.body.setAwake( false );
                    } else {
                        skeleton.dontPutToSleep = true;
```

```
                    }
                } else {
                    skeleton.setSkeletonEntitiesToSleepRecursively( );
                    skeleton.body.setActive( true );
                    skeleton.body.setAwake( false );
                }
            }
        }
    }

    public boolean inRectangleBounds( Rectangle rect, Vector2 point ) {
        if ( point.x > rect.x && point.x < rect.x + rect.width
                && point.y > rect.y && point.y < rect.y + rect.height ) {
            return true;
        }
        return false;
    }

    public boolean isRemoved( ) {
        return removed;
    }

    /**
     * This update function is ONLY called on the very root skeleton, it takes
     * care of the child sksletons
     *
     * @author stew
     */
    @Override
    public void update( float deltaTime ) {
        if ( this.getPositionPixel( ).y < MAX_FALL_POS && !this.removed ) {
            this.remove( );
        } else {
            if ( !removed ) {
                if ( !this.removeNextStep ) {
                    super.update( deltaTime );
                    float frameRate = 1 / deltaTime;
                    isUpdatable = ( !this.isFadingSkel( ) || this.isFGFaded( ) )
                            || this.dontPutToSleep;
                    if ( useBoundingRect && updatedOnce ) {
                        boundingRect.x = this.getPositionPixel( ).x
                                - ( boundingRect.width / 2.0f );
                        boundingRect.y = this.getPositionPixel( ).y
                                - ( boundingRect.height / 2.0f );
                        if ( !boundingRect.overlaps( lastCameraRect ) ) {
```



```
                            isUpdatable = false;
                            if ( !wasInactive ) {
                                wasInactive = true;
                                setSkeletonEntitiesToSleepRecursively( );
                            }
                        } else {
                            isUpdatable = true;
                        }
                    } else if ( !useBoundingRect && !isUpdatable
                            && this.setChildSkeletonsToSleep && !wasInactive ) {
                        setSkeletonEntitiesToSleepRecursively( );
                    }
                    updatedOnce = true;
                    if ( isUpdatable || isMacroSkeleton ) {
                        updateMover( deltaTime );
                        if ( entityType != EntityType.ROOTSKELETON
                                && isKinematic( ) ) {
                            super.setTargetPosRotFromSkeleton( frameRate,
                                    parentSkeleton );
                        }
                    }
                    for ( EventTrigger event : eventMap.values( ) ) {
                        event.translatePosRotFromSKeleton( this );
                        // event.setTargetPosRotFromSkeleton( frameRate, this );
                    }

                    if ( isUpdatable ) {
                        for ( Rope rope : ropeMap.values( ) ) {
                            // TODO: ropes need to be able to be deleted
                            if ( wasInactive ) {
                                boolean nextLink = true;
                                int index = 0;
                                if ( rope.getEndAttachment( ) != null ) {
                                    if ( !rope.getEndAttachment( ).body
                                            .isActive( ) ) {
                                        rope.getEndAttachment( ).body
                                                .setActive( true );
                                    }
                                    // if ( rope.getEndAttachment(
                                    // ).body.isAwake( ) ) {
                                    // rope.getEndAttachment( ).body.setAwake(
                                    // false );
                                    // }
                                }
                                while ( nextLink ) {
```

```
                    if ( !rope.getLink( index ).body.isActive( ) ) {
                        rope.getLink( index ).body
                                .setActive( true );
                    }
                    // if ( rope.getLink( index ).body.isAwake(
                    // ) ) {
                    // rope.getLink( index ).body.setAwake(
                    // false );
                    // }
                    if ( rope.getLastLink( ) == rope
                            .getLink( index ) ) {
                        nextLink = false;
                    }
                    index++;
                }
            }
            rope.update( deltaTime );
        }
        for ( Platform platform : kinematicPlatformMap.values( ) ) {
            if ( platform.removeNextStep ) {
                entitiesToRemove.add( platform );
            } else {
                if ( wasInactive ) {
                    if ( !platform.body.isActive( ) ) {
                        platform.body.setActive( true );
                    }
                    if ( platform.body.isAwake( ) ) {
                        platform.body.setAwake( false );
                    }
                    platform.translatePosRotFromSKeleton( this );
                    platform.update( deltaTime );
                } else {
                    platform.updateMover( deltaTime );
                    if ( !platform.body.isActive( ) ) {
                        platform.body.setActive( true );
                    }
                    if ( platform.body.isAwake( ) ) {
                        platform.body.setAwake( false );
                    }
                    if ( platform.hasMoved( )
                            || platform.hasRotated( )
                            || hasMoved( ) || hasRotated( ) ) {
                        platform.setTargetPosRotFromSkeleton(
                                frameRate, this );
                        platform.setPreviousTransformation( );
```



```
                    } else {
                        platform.body
                                .setLinearVelocity( Vector2.Zero );
                        platform.body.setAngularVelocity( 0.0f );
                    }
                    platform.update( deltaTime );
                }
            }
        }
        for ( Platform platform : dynamicPlatformMap.values( ) ) {
            if ( platform.removeNextStep ) {
                entitiesToRemove.add( platform );
            } else {
                if ( wasInactive ) {
                    if ( !platform.body.isActive( ) ) {
                        platform.body.setActive( true );
                    }
                    if ( platform.body.isAwake( ) ) {
                        platform.body.setAwake( false );
                    }
                }
                platform.updateMover( deltaTime );
                platform.update( deltaTime );
            }
        }
        for ( CheckPoint chkpt : checkpointMap.values( ) ) {
            if ( chkpt.removeNextStep ) {
                entitiesToRemove.add( chkpt );
            } else {
                if ( wasInactive ) {
                    if ( !chkpt.body.isActive( ) ) {
                        chkpt.body.setActive( true );
                    }
                    if ( chkpt.body.isAwake( ) ) {
                        chkpt.body.setAwake( false );
                    }
                }
                chkpt.update( deltaTime );
            }
        }
        for ( Screw screw : screwMap.values( ) ) {
            if ( screw.removeNextStep ) {
                entitiesToRemove.add( screw );
            } else {
                if ( wasInactive ) {
```

```
                        if ( !screw.body.isActive( ) ) {
                            screw.body.setActive( true );
                        }
                        if ( screw.body.isAwake( ) ) {
                            screw.body.setAwake( false );
                        }
                    }
                    screw.update( deltaTime );
                }
            }
            if ( wasInactive ) {
                if ( !body.isActive( ) ) {
                    body.setActive( true );
                }
                if ( body.isAwake( ) ) {
                    body.setAwake( false );
                }
                for ( Skeleton skeleton : childSkeletonMap.values( ) ) {
                    if ( !skeleton.body.isActive( ) ) {
                        skeleton.body.setActive( true );
                    }
                    if ( skeleton.body.isAwake( ) ) {
                        skeleton.body.setAwake( false );
                    }
                }
                wasInactive = false;
            }
        } else {
            if ( !wasInactive ) {
                setEntitiesToSleepOnUpdate( );
                wasInactive = true;
            }
        }

        setPreviousTransformation( );

        alphaFadeAnimator.update( deltaTime );
        Vector2 pixelPos = null;
        if ( fgSprite != null ) {
            pixelPos = getPosition( ).mul( Util.BOX_TO_PIXEL );
            fgSprite.setPosition( pixelPos.x - offset.x, pixelPos.y
                    - offset.y );
            fgSprite.setRotation( MathUtils.radiansToDegrees
                    * getAngle( ) );
        }
```



```
        if ( bgSprite != null ) {
            if ( pixelPos == null )
                pixelPos = getPosition( ).mul( Util.BOX_TO_PIXEL );
            bgSprite.setPosition( pixelPos.x - offset.x, pixelPos.y
                    - offset.y );
            bgSprite.setRotation( MathUtils.radiansToDegrees
                    * getAngle( ) );
        }
        updateDecals( deltaTime );

        // }
        // recursively update child skeletons

        for ( Skeleton skeleton : childSkeletonMap.values( ) ) {
            if ( skeleton.removeNextStep ) {
                entitiesToRemove.add( skeleton );
            } else {
                if ( !setChildSkeletonsToSleep || isUpdatable
                        || skeleton.dontPutToSleep ) {
                    skeleton.update( deltaTime );
                }
            }
        }

        // remove stuff
        if ( entitiesToRemove.size( ) > 0 ) {

            for ( Entity e : entitiesToRemove ) {

                switch ( e.entityType ) {
                case SKELETON:
                    Skeleton s = childSkeletonMap.remove( e.name );
                    s.remove( );
                    break;
                case PLATFORM:
                    Platform p;
                    if ( e.isKinematic( ) ) {
                        p = kinematicPlatformMap.remove( e.name );
                    } else {
                        p = dynamicPlatformMap.remove( e.name );
                    }
                    p.remove( );
                    break;
                case SCREW:
                    Screw sc = screwMap.remove( e.name );
```

```
                                sc.remove( );
                                break;
                            case CHECKPOINT:
                                CheckPoint chkpt = checkpointMap
                                        .remove( e.name );
                                chkpt.setNextCheckPointInPM( );
                                chkpt.remove( );
                                break;
                            default:
                                throw new RuntimeException(
                                        "You are trying to remove enity '"
                                                + e.name
                                                + "' but skeleton '"
                                                + this.name
                                                + "' can't determine it's type.
This may be my fault for not adding a case. -stew" );
                            }
                        }
                        entitiesToRemove.clear( );
                    }
                }
            }
        }
    }

    /**
     * removes the bodies and joints of all the skeletons children
     */
    @Override
    public void remove( ) {
        for ( Skeleton skeleton : childSkeletonMap.values( ) ) {
            skeleton.remove( );
        }
        childSkeletonMap.clear( );
        for ( Platform p : dynamicPlatformMap.values( ) ) {
            p.remove( );
        }
        dynamicPlatformMap.clear( );
        for ( Platform p : kinematicPlatformMap.values( ) ) {
            p.remove( );
        }
        kinematicPlatformMap.clear( );
        for ( Screw screw : screwMap.values( ) ) {
            screw.remove( );
        }
```



```
        screwMap.clear( );
        for ( CheckPoint chkpt : checkpointMap.values( ) ) {
            chkpt.setNextCheckPointInPM( );
            chkpt.remove( );
        }
        checkpointMap.clear( );
        for ( EventTrigger event : eventMap.values( ) ) {
            event.remove( );
        }
        eventMap.clear( );
        // for ( Rope rope : ropeMap.values( ) ) {
        // boolean nextLink = true;
        // int index = 0;
        // if ( rope.getEndAttachment( ) != null ) {
        // while ( rope.getEndAttachment( ).body.getJointList( ).iterator(
        // ).hasNext( ) ) {
        // world.destroyJoint( body.getJointList( ).get( 0 ).joint );
        // }
        // world.destroyBody( rope.getEndAttachment( ).body );
        // }
        // while ( nextLink ) {
        // world.destroyBody( rope.getLink( index ).body );
        // if ( rope.getLastLink( ) == rope.getLink( index ) ) {
        // nextLink = false;
        // }
        // index++;
        // }
        // }
        // while ( body.getJointList( ).iterator( ).hasNext( ) ) {
        // world.destroyJoint( body.getJointList( ).get( 0 ).joint );
        // }
        body.setActive( false );
        body.setAwake( true );
        // world.destroyBody( body );
        // this.fgDecals.clear( );
        // this.bgDecals.clear( );
        // this.bgSprite = null;
        // this.fgSprite = null;
        this.removed = true;
    }

    /**
     * this skeleton has gone to bed, put its entities to sleep instead of
     * updating the entities movements and such and delete them if necessary
     */
```

```
        private void setEntitiesToSleepOnUpdate( ) {
            if ( !this.removeNextStep ) {
                for ( Platform platform : kinematicPlatformMap.values( ) ) {
                    if ( platform.removeNextStep ) {
                        entitiesToRemove.add( platform );
                    } else if ( !platform.dontPutToSleep ) {
                        platform.body.setAwake( true );
                        platform.body.setActive( false );
                    }
                }
                for ( Platform platform : dynamicPlatformMap.values( ) ) {
                    if ( platform.removeNextStep ) {
                        entitiesToRemove.add( platform );
                    } else {
                        platform.body.setAwake( true );
                        platform.body.setActive( false );
                    }
                }
                for ( CheckPoint chkpt : checkpointMap.values( ) ) {
                    if ( chkpt.removeNextStep ) {
                        entitiesToRemove.add( chkpt );
                    } else {
                        chkpt.body.setActive( true );
                        chkpt.body.setAwake( false );
                    }
                }
                for ( Screw screw : screwMap.values( ) ) {
                    if ( screw.removeNextStep ) {
                        entitiesToRemove.add( screw );
                    } else if ( !screw.dontPutToSleep ) {
                        if ( this.useBoundingRect ) {
                            if ( inRectangleBounds( this.boundingRect,
                                    screw.getPositionPixel( ) ) ) {
                                if ( screw.getDepth( ) >= 0 ) {
                                    screw.body.setAwake( true );
                                    screw.body.setActive( false );
                                } else {
                                    screw.dontPutToSleep = true;
                                }
                            } else {
                                screw.dontPutToSleep = true;
                            }
                        } else {
                            screw.body.setAwake( true );
                            screw.body.setActive( false );
```



```
                        }
                    }
                }
                for ( Rope rope : ropeMap.values( ) ) {
                    // TODO: ropes need to be able to be deleted
                    boolean nextLink = true;
                    int index = 0;
                    if ( rope.getEndAttachment( ) != null ) {
                        // rope.getEndAttachment( ).body.setAwake( true );
                        rope.getEndAttachment( ).body.setActive( false );
                    }
                    while ( nextLink ) {
                        // rope.getLink( index ).body.setAwake( true );
                        rope.getLink( index ).body.setActive( false );
                        if ( rope.getLastLink( ) == rope.getLink( index ) ) {
                            nextLink = false;
                        }
                        index++;
                    }
                }
            }
        }

        /**
         *
         * @param batch
         * @param camera
         */
        @Override
        public void drawFGDecals( SpriteBatch batch, Camera camera ) {
            if ( !removed && !removeNextStep ) {
                for ( Sprite decal : fgDecals ) {
                    if ( decal.alpha >= 0.25 ) {
                        if ( decal.getBoundingRectangle( ).overlaps(
                                camera.getBounds( ) ) ) {
                            decal.draw( batch );
                        }
                    }
                }
            }
        }

        @Override
        public void draw( SpriteBatch batch, float deltaTime, Camera camera ) {
            if ( !removed && !removeNextStep ) {
```

```
            // if ( this.useBoundingRect ) {
            // shapeRender.setProjectionMatrix( camera.combined( ) );
            // shapeRender.begin( ShapeType.Rectangle );
            // shapeRender.rect( boundingRect.x, boundingRect.y,
            // boundingRect.width,
            // boundingRect.height );
            // shapeRender.end( );
            // }
            super.draw( batch, deltaTime, camera );
            if ( visible ) {
                drawChildren( batch, deltaTime, camera );
                if ( fgSprite != null && alphaFadeAnimator.getTime( ) > 0 ) {
                    fgSprite.setAlpha( alphaFadeAnimator.getTime( ) );
                    // batch.setColor( c.r, c.g, c.b, fgAlphaAnimator.getTime( )
                    // );
                    // fgSprite.draw( batch );
                    // batch.setColor( c.r, c.g, c.b, oldAlpha );
                }
                if ( applyFadeToFGDecals ) {
                    if ( name.equals( "head_skeleton" ) )
                        getAngle( );
                    fadeFGDecals( );
                }
            }
        }
    }

    private void drawChildren( SpriteBatch batch, float deltaTime, Camera camera ) {
        if ( !removed && !removeNextStep ) {
            lastCameraRect = camera.getBounds( );
            if ( !wasInactive && isUpdatable ) {
                for ( EventTrigger et : eventMap.values( ) ) {
                    et.draw( batch, deltaTime, camera );
                }
                for ( Screw screw : screwMap.values( ) ) {
                    if ( !screw.getRemoveNextStep( ) ) {
                        screw.draw( batch, deltaTime, camera );
                    }
                }
                for ( Platform p : dynamicPlatformMap.values( ) ) {
                    drawPlatform( p, batch, deltaTime, camera );
                }
                for ( Platform p : kinematicPlatformMap.values( ) ) {
                    drawPlatform( p, batch, deltaTime, camera );
                }
```



```
                for ( CheckPoint chkpt : checkpointMap.values( ) ) {
                    if ( !chkpt.getRemoveNextStep( ) ) {
                        chkpt.draw( batch, deltaTime, camera );
                    }
                }
                for ( Rope rope : ropeMap.values( ) ) {
                    rope.draw( batch, deltaTime, camera );
                }
            }
            // draw the entities of the parent skeleton before recursing through
            // the
            // child skeletons
            // if ( isUpdatable || isMacroSkeleton )
            {
                for ( Skeleton skeleton : childSkeletonMap.values( ) ) {
                    if ( !setChildSkeletonsToSleep || isUpdatable
                            || skeleton.dontPutToSleep ) {
                        skeleton.draw( batch, deltaTime, camera );
                    }
                }
            }
        }
    }

    /**
     *
     * @param batch
     * @param camera
     */
    @Override
    public void drawBGDecals( SpriteBatch batch, Camera camera ) {
        if ( !removed && !removeNextStep ) {
            for ( Sprite decal : bgDecals ) {
                if ( decal.getBoundingRectangle( )
                        .overlaps( camera.getBounds( ) ) ) {
                    if ( !invisibleBGDecal ) {
                        decal.draw( batch );
                    }
                }
            }
        }
    }

    /**
     * Draw each child. Tiled platforms have unique draw calls. Platforms can be
```

```
     * hazards as well
     */
    private void drawPlatform( Platform platform, SpriteBatch batch,
            float deltaTime, Camera camera ) {
        platform.draw( batch, deltaTime, camera );
    }

    public boolean getWasInactive( ) {
        return wasInactive;
    }

    public void setUseBoundingRect( boolean setting ) {
        useBoundingRect = setting;
    }

    public boolean getIsUsingBoundingBox( ) {
        return useBoundingRect;
    }

    public boolean isUpdatable( ) {
        return isUpdatable;
    }

    private String getUniqueName( String nonUniqueName ) {
        return nonUniqueName + "-NON-UNIQUE-NAME_" + entityCount;
    }

    /**
     * Delete a child skeleton by name. Recursively tries to find the child
     * skele.
     *
     * @param skeleName
     *            searches all skeletons under this skeleton
     */
    public void deleteSkeletonByName( String skeleName ) {
        for ( Skeleton s : childSkeletonMap.values( ) ) {
            if ( s.name.equals( skeleName ) ) {
                rootSkeleton.destroySkeleton( s );
                break;
            } else {
                s.deleteSkeletonByName( skeleName );
            }
        }
    }
```



```
    /**
     * Deletes this skeleton, Potentially creates null pointers, please don't
     * directly call this, instead add your skeleton-to-be-deleted to root using
     * RootSkeleton.deleteSkeleton(Skeleton)
     */
    @Override
    public void dispose( ) {
        for ( Platform platform : dynamicPlatformMap.values( ) ) {
            platform.body.getWorld( ).destroyBody( platform.body );
        }
        dynamicPlatformMap.clear( );
        for ( Platform platform : kinematicPlatformMap.values( ) ) {
            platform.body.getWorld( ).destroyBody( platform.body );
        }
        kinematicPlatformMap.clear( );
        for ( Rope rope : ropeMap.values( ) ) {
            rope.dispose( );
        }
        ropeMap.clear( );
        for ( Screw screw : screwMap.values( ) ) {
            screw.dispose( );
        }
        for ( CheckPoint chkpt : checkpointMap.values( ) ) {
            chkpt.dispose( );
        }
        screwMap.clear( );
        for ( EventTrigger et : eventMap.values( ) ) {
            et.dispose( );
        }
        eventMap.clear( );
        for ( CheckPoint chkpt : checkpointMap.values( ) ) {
            chkpt.dispose( );
        }
        checkpointMap.clear( );
        super.dispose( );
    }

    /**
     * Generally for debug purposes
     *
     * @param angleInRadians
     */
    public void rotateBy( float angleInRadians ) {
        setLocalRot( getLocalRot( ) + angleInRadians );
    }
```

```
    public void setChildSkeletonsToSleepProperty( boolean setting ) {
        setChildSkeletonsToSleep = setting;
    }

    /**
     * For debugging
     *
     * @param xPixel
     * @param yPixel
     */
    public void translateBy( float xPixel, float yPixel ) {
        setLocalPos( getLocalPos( ).add( xPixel, yPixel ) );
    }

    /**
     * A less recursive get root function!
     *
     * @return Root skeleton of this skeleton
     */
    public RootSkeleton getRoot( ) {
        return rootSkeleton;
    }

    /**
     *
     * @param hasTransparency
     *            true if you want to see into the robot
     */
    public void setFade( boolean hasTransparency ) {
        float speed = fadeSpeed;
        // if ( !hasTransparency ){
        // Gdx.app.log("stageSkeleton","NO TRANSPARENCY");
        // }
        if ( hasTransparency ) {
            speed = -fadeSpeed;
        }
        /*
         * else{ if(name.equals("stageSkeleton")){
         *
         * //speed = fadeSpeed; } } if(name.equals("stageSkeleton"))
         * Gdx.app.log(
         * "stageSkeleton","Speed: "+speed+" Time:"+alphaFadeAnimator.getTime(
         * ));
         */
```



```
            alphaFadeAnimator.speed( speed );
        }

        private void fadeFGDecals( ) {
            float alpha = alphaFadeAnimator.getTime( );
            alpha *= alpha;
            for ( Sprite decal : fgDecals ) {
                if ( decal.getAlpha( ) != alpha ) {
                    decal.setAlpha( alpha );
                }
            }
        }

        public void setFgFade( boolean applyFadeToFGDecals ) {
            this.applyFadeToFGDecals = applyFadeToFGDecals;
        }

        public boolean isFGFaded( ) {
            return alphaFadeAnimator.getTime( ) < 1;
        }

        public boolean isFadingSkel( ) {
            return applyFadeToFGDecals;
        }

        public EventTrigger getEvent( String eventName ) {
            return eventMap.get( eventName );
        }
    }
```

```java
package com.blindtigergames.werescrewed.entity.builders;

import com.badlogic.gdx.graphics.Texture;
// [omitted]
import com.blindtigergames.werescrewed.eventTrigger.EventTrigger;

public class SkeletonBuilder extends GenericEntityBuilder< SkeletonBuilder > {

    protected Array< Vector2 > polyVertsFG, polyVertsBG, invisibleVerts;

    protected float density;
    private BodyType bodyType;
    protected boolean onBGverts;
    protected Texture texBackground, texForeground, texBody;
    protected boolean hasDeactivateTrigger;
    protected boolean fadeFgDecals;
    protected boolean setChildSkeletonsToSleep = false;
    protected boolean useBoundingRect = false;
    protected Rectangle boundingRect;

    protected boolean lessExtraBorder = false;

    public SkeletonBuilder( World world ) {
        super( );
        reset( );
        super.world = world;
    }

    @Override
    public SkeletonBuilder reset( ) {
        super.reset( );
        this.polyVertsFG = null;
        this.polyVertsBG = null;
        this.bodyType = BodyType.KinematicBody;
        this.density = 1.0f;
        this.onBGverts = true;
        // background textures
        this.texBackground = WereScrewedGame.manager.getLevelRobotBGTex( );
        this.texForeground = WereScrewedGame.manager.getLevelRobotFGTex( );
        this.texBody = null;
        this.hasDeactivateTrigger = false;
        this.fadeFgDecals = false;
        this.invisibleVerts = null;
        this.setChildSkeletonsToSleep = false;
        return this;
```



```java
    }

    /**
     * All following verts added will set to the background polysprite of this
     * skeleton This is true by default
     *
     * @return
     */
    public SkeletonBuilder bg( ) {
        this.onBGverts = true;
        return this;
    }

    public SkeletonBuilder lessExtraBorder( ) {
        this.lessExtraBorder = true;
        return this;
    }

    /**
     * All following verts will apply to the foreground polysprite
     *
     * @return
     */
    public SkeletonBuilder fg( ) {
        this.onBGverts = false;
        return this;
    }

    public SkeletonBuilder hasDeactiveTrigger( boolean hasTrigger ) {
        this.hasDeactivateTrigger = hasTrigger;
        return this;
    }

    public SkeletonBuilder texForeground( Texture fgTex ) {
        this.texForeground = fgTex;
        return this;
    }

    public SkeletonBuilder texBackground( Texture bgTex ) {
        this.texBackground = bgTex;
        return this;
    }

    public SkeletonBuilder texBody( Texture bodyTex ) {
        this.texBody = bodyTex;
```

```
        return this;
    }

    public SkeletonBuilder setUseBoundingRect( boolean setting ) {
        useBoundingRect = setting;
        return this;
    }

    public SkeletonBuilder buildRectangle( float x, float y, float width, float
height ) {
        boundingRect = new Rectangle( x, y, width, height);
        return this;
    }

    /**
     * Set the entire vertice list for the polySprite on the next built skeleton
     *
     * @param verts
     *            array of verts in pixels.
     * @return
     */
    public SkeletonBuilder setVerts( Array< Vector2 > verts ) {
        if ( onBGverts ) {
            this.polyVertsBG = verts;
        } else {
            this.polyVertsFG = verts;
        }
        return this;
    }

    public SkeletonBuilder invisibleVerts( Array< Vector2 > verts ) {
        this.invisibleVerts = verts;
        return this;
    }

    public SkeletonBuilder setChildSkelsToSleep ( boolean setting ) {
        setChildSkeletonsToSleep = setting;
        return this;
    }

    /**
     * Add a vertice to the polySprite for this skeleton
     *
     * @param vert
     *            , (x,y) in pixels
```



```
     * @return
     */
    public SkeletonBuilder vert( Vector2 vert ) {
        Array< Vector2 > vertList;
        if ( onBGverts ) {
            if ( polyVertsBG == null ) {
                polyVertsBG = new Array< Vector2 >( );
            }
            vertList = polyVertsBG;
        } else {
            if ( polyVertsFG == null ) {
                polyVertsFG = new Array< Vector2 >( );
            }
            vertList = polyVertsFG;
        }
        vertList.add( vert );
        return this;
    }

    /**
     * Add a vertice to the polySprite for this skeleton
     *
     * @param x
     *            x-position in pixels
     * @param y
     *            y-position in pixels.
     * @return
     */
    public SkeletonBuilder vert( float x, float y ) {
        return this.vert( new Vector2( x, y ) );
    }

    public SkeletonBuilder dynamic( boolean d ) {
        if ( d ) {
            return this.dynamic( );
        }
        return this.kinematic( );
    }

    public SkeletonBuilder dynamic( ) {
        bodyType = BodyType.DynamicBody;
        return this;
    }

    public SkeletonBuilder staticBody( ) {
```

```
            bodyType = BodyType.StaticBody;
            return this;
        }

        public SkeletonBuilder kinematic( ) {
            bodyType = BodyType.KinematicBody;
            return this;
        }

        public SkeletonBuilder fadeFgDecals( boolean applyFadeToFgDecals ) {
            this.fadeFgDecals = applyFadeToFgDecals;
            return this;
        }

        /**
         *
         * @param density
         *            - float used for density, default is 1.0f
         * @return SkeletonBuilder
         */
        public SkeletonBuilder density( float density ) {
            this.density = density;
            return this;
        }



        /**
         * Builds a friggin root skeleton, what do you want jeese.
         */
        public RootSkeleton buildRoot( ) {
            return new RootSkeleton( "root", new Vector2( ), null, world );
        }

        @Override
        public Skeleton build( ) {
            Skeleton out = new Skeleton( name, pos, null, super.world, bodyType );
            out.setChildSkeletonsToSleepProperty( setChildSkeletonsToSleep );
            out.setUseBoundingRect( useBoundingRect );
            out.boundingRect = this.boundingRect;
            if ( invisibleVerts != null ) {
                if ( polyVertsFG != null && texForeground != null ) {
                    out.fgSprite = new PolySprite( texForeground, polyVertsFG );
                }
            }
```



```
            if ( polyVertsBG != null && texBackground != null ) {
                out.bgSprite = new PolySprite( texBackground, polyVertsBG );
            }

            // out.body.setType( bodyType );
            out.setDensity( this.density );

            if ( invisibleVerts != null ) {
                EventTriggerBuilder etb = new EventTriggerBuilder( world );
                etb.name( name + "-invisible-fader" ).setVerts( invisibleVerts );

                if(this.lessExtraBorder)
                    etb.extraBorder( 256f );
                else
                    etb.extraBorder( 300f );


                EventTrigger et = etb.position( pos.add( 0, 0 ) ).addEntity( out )
                        .beginAction( new FadeSkeletonAction( true ) )
                        .endAction( new FadeSkeletonAction( false ) ).repeatable( )
                        .twoPlayersToDeactivate( ).build( );

                out.addEventTrigger( et );
            } else {
                // PIZZA
                if ( hasDeactivateTrigger && polyVertsBG != null ) {
                    EventTriggerBuilder etb = new EventTriggerBuilder( world );
                    etb.name( name + "-activator" ).setVerts( polyVertsBG );


                    if(this.lessExtraBorder)
                        etb.extraBorder( 128f );
                    else
                        etb.extraBorder( 300f );

                    EventTrigger et = etb.position( pos ).addEntity( out )
                            .beginAction( new FadeSkeletonAction( true ) )
                            .endAction( new FadeSkeletonAction( false ) )
                            .repeatable( ).twoPlayersToDeactivate( ).build( );
                    out.addEventTrigger( et );
                        // Gdx.app.log( "SkeletonBuilder",
                        // "I just built an event trigger" );
                    } else if ( polyVertsFG != null ) {
                        EventTriggerBuilder etb = new EventTriggerBuilder( world );
```

```
                    etb.name( name + "-fg-fader" ).setVerts( polyVertsFG );


                if(this.lessExtraBorder)
                    etb.extraBorder( 128f );
                else
                    etb.extraBorder( 300f );

                    EventTrigger et = etb.position( pos.add( 0, 0 ) ).addEntity( out )
                            .beginAction( new FadeSkeletonAction( true ) )
                            .endAction( new FadeSkeletonAction( false ) )
                            .repeatable( ).twoPlayersToDeactivate( ).build( );
                    out.addEventTrigger( et );
                }
            }

            if ( fadeFgDecals ) {
                out.setFgFade( fadeFgDecals );
            }

            return out;
        }
    }
```

```lua
--A heap that sorts keys in a {key, value} pair set of data
-- such that keys are type(number)

local Heap = {}
Heap.mt = {} --metatable
Heap.prototype = {}
Heap.mt.__index = Heap.prototype

-- Utility Function
table.exchange = function(t, a, b)
    local tmp = t[a]
    t[a] = t[b]
    t[b] = tmp
end

--data := { { k = type(number), v = (Anything) }, ... }
function Heap.new(isMax,data)
    data = data or {}
    local heap = {}
    setmetatable(heap, Heap.mt)
    heap.isMax = isMax and true
    heap.heapsize = #data
    for i = 1, #data do
        heap[i] = data[i]
    end
    return heap
end

function Heap.prototype.parent(self,i)
    return math.max(1,bit32 and bit32.rshift(i,1) or math.floor(i/2))
end

function Heap.prototype.left(self,i)
    return math.max(1,bit32 and bit32.lshift(i,1) or 2*i)
end

function Heap.prototype.right(self,i)
    return math.max(1,bit32 and (bit32.lshift(i,1)+1) or (2*i+1))
end

function Heap.prototype.minHeapify(self,i)
    if self.isMax == true then error("Can't call minHeapify on max heap") end
    local l = self:left(i)
    local r = self:right(i)
    local smallest = nil
```



```lua
    if l <= self.heapsize and self[l].k < self[i].k then
        smallest = l
    else
        smallest = i
    end
    if r <= self.heapsize and self[r].k < self[smallest].k then
        smallest = r
    end
    if smallest ~= i then
        table.exchange(self,i,smallest)
        self.minHeapify(self,smallest)
    end
end

function Heap.prototype.buildMinHeap(self)
    self.isMax = false
    self.heapsize = #self
    for i = math.floor(#self), 1, -1 do
        self.minHeapify(self,i)
    end
end

function Heap.prototype.maxHeapify(self,i)
    if self.isMax ~= true then error("Can't call maxHeapify on min heap") end
    local l = self:left(i)
    local r = self:right(i)
    local largest = nil
    if l <= self.heapsize and self[l].k > self[i].k then
        largest = l
    else
        largest = i
    end
    if r <= self.heapsize and self[r].k > self[largest].k then
        largest = r
    end
    if largest ~= i then
        table.exchange(self,i,largest)
        self.maxHeapify(self,largest)
    end
end

function Heap.prototype.buildMaxHeap(self)
    self.isMax = true
    self.heapsize = #self
    for i = math.floor(#self), 1, -1 do
```

```
            self.maxHeapify(self,i)
        end
    end

    function Heap.prototype.heapsort(self)
        local heapify = nil
        if self.isMax == true then
            heapify = self.maxHeapify
            self.buildMaxHeap(self)
        else
            heapify = self.minHeapify
            self.buildMinHeap(self)
        end
        for i = #self, 2, -1 do
            table.exchange(self,1,i)
            self.heapsize = self.heapsize-1
            heapify(self,1)
        end
    end

    function Heap.prototype.size(self)
        return self.heapsize
    end

    ------------------------------------------------------
    -- Priority Queue methods
    ------------------------------------------------------
    -- Max priority queue methods
    function Heap.prototype.maximum(self,i)
        if self.isMax ~= true then
            --no maximum garanteed when using a min priority queue
            error("maximum(): invalid operation on min heap.")
        end
        return self[1]
    end

    function Heap.prototype.extractMax(self)
        if self.isMax ~= true then
            error("extractMax(): invalid operation on min heap.")
        end
        if self.heapsize < 1 then error("extractMax(): heap underflow") end
        local max = self[1]
        self[1] = self[self.heapsize]
        self.heapsize = self.heapsize - 1
        self:maxHeapify(1)
```



```
        return max
    end

    function Heap.prototype.increaseKey(self,i,key)
        if self.isMax ~= true then
            error("increaseKey(): invalid operation on min heap.")
        end
        if key < self[i].k then
            error "new key is smaller than current key"
        end
        self[i].k = key
        while i > 1 and self[self:parent(i)].k < self[i].k do
            table.exchange(self, i, self:parent(i))
            i = self:parent(i)
        end
    end

    --
    -- Min priority queue methods
    --
    function Heap.prototype.minimum(self)
        if self.isMax == true then
            --no maximum garanteed when using a min priority queue
            error("minimum(): invalid operation on max heap.")
        end
        return self[1]
    end

    function Heap.prototype.extractMin(self,i)
        if self.isMax == true then
            error("extractMin(): invalid operation on max heap.")
        end
        if self.heapsize < 1 then error("extractMin(): heap underflow") end
        local min = self[1]
        self[1] = self[self.heapsize]
        self.heapsize = self.heapsize - 1
        self:minHeapify(1)
        return min
    end

    function Heap.prototype.decreaseKey(self,i,key)
        if self.isMax == true then
            error("decreaseKey(): invalid operation on max heap.")
        end
        if key > self[i].k then
```

```
            error "new key is bigger than current key"
        end
        self[i].k = key
        while i > 1 and self[self:parent(i)].k > self[i].k do
            table.exchange(self, i, self:parent(i))
            i = self:parent(i)
        end
    end

    --
    -- Standard priority queue methods
    --
    function Heap.prototype.insert(self,key,value)
        self.heapsize = self.heapsize+1
        if self.isMax then
            self[self.heapsize] = {k = -math.huge, v = value}
            self:increaseKey(self.heapsize,key)
        else
            self[self.heapsize] = {k = math.huge, v = value}
            self:decreaseKey(self.heapsize,key)
        end
    end

    function Heap.prototype.removeKey(self,key)
        return self:remove("k",key)
    end

    function Heap.prototype.removeValue(self,value)
        return self:remove("v",value)
    end

    --Removes first node found with given key/value
    function Heap.prototype.remove(self, kOrV, obj)
        assert(kOrV=="k" or kOrV=="v", [[Heap:remove() generalizes pairs by 'k' or 'v',
     therefore you must use one of these as the first parameter of find().]])
        local index, pair = self:find(kOrV,obj)
        if index then
            self[index] = self[self.heapsize]
            self.heapsize = self.heapsize - 1
            local heapify = self.isMax and self.maxHeapify or self.minHeapify
            heapify(self,index)
            return pair
        end
        return nil
    end
```



```
    --Updates the first found [k,v] with the new key
    function Heap.prototype.updateKeyByValue(self,value,newKey)
        self:removeValue(value)
        self:insert(newKey,value)
    end

    function Heap.prototype.findKey(self,key)
        return self:find("k",key)
    end

    function Heap.prototype.findValue(self,value)
        return self:find("v",value)
    end

    --Returns the index and pair of the first k or v object you're looking for
    function Heap.prototype.find(self,kOrV,obj)
        assert(kOrV=="k" or kOrV=="v", [[Heap:find() generalizes pairs by 'k' or 'v',
     therefore you must use one of these as the first parameter of find().]])
        for i=1, self.heapsize do
            if self[i][kOrV] == obj then
                return i, self[i]
            end
        end
        return nil, nil
    end

    -- Debug Utility methods
    function Heap.prototype.print(self,m)
        local out = (m and (m..': ') or '') .. ('['..(self[1].k or '')..', '..
    (self[1].v or '') ..']')
        for i = 2, self.heapsize do
            out = out .. ', ' .. '['.. self[i].k .. ', ' .. self[1].v .. ']'
        end
        out = out .. ' {'
        for i = self.heapsize+1, #self do
            out = out .. ', ' .. '['.. self[i].k .. ', ' .. self[1].v .. ']'
        end
        out = out .. ' }'
        print(out)
    end

    return Heap
```

```lua
-----------------------------------------------------------------------------
-- calllog.lua
-- by Stewart Bracken  http://stewart.bracken.bz  stew.bracken@gmail.com
-- List call logs from Twilio with Corona widgets.
-- Lots of code leveraged from WidgetDemo.
-- Known bug: if you switch to another tab before if finished retriving calls,
-- this get funky.
-- Known critical bug: text labels and buttons are recreated on every scene
-- enter but never destroyed, adding up a lot of widgets and whatnot.
-----------------------------------------------------------------------------

local widget = require( "widget" )
local storyboard = require( "storyboard" )
local scene = storyboard.newScene()

-- create a constant for the left spacing of the row content
local LEFT_PADDING = 10

local tableView = nil

local isExiting = false

function scene:createTableLog(twilio_response)
    -- Forward reference for our tableView
    local group = self.view

    --Data to be displayed when a row is selected
    local rowDisplayData = {['To:'] = 'to_formatted', ['From:'] = 'from_formatted',
['Status:'] = 'status', ['Duration:'] = 'duration'}

    -- Text to show which item we selected
    local itemSelected = {}
    local itemY = 150
    for k, v in pairs(rowDisplayData) do
       itemSelected[k] = display.newText( k, 0, 0, native.systemFontBold, 14 )
       itemSelected[k]:setTextColor( 0 )
       itemSelected[k].x = display.contentWidth + itemSelected[k].contentWidth * 0.5
       itemSelected[k].y = itemY
       group:insert( itemSelected[k] )
       itemY = itemY + itemSelected[k].contentHeight + 10
    end
    local selectedItemAnchor = itemSelected['To:']

    -- Function to return to the list
    local function goBack( event )
```



```lua
        --Transition in the list, transition out the item selected text and the
back button
        transition.to( tableView, { x = 0, time = 400, transition = easing.outExpo
} )

        for k, v in pairs(itemSelected) do
            local item = v
            transition.to( item, { x = display.contentWidth + item.contentWidth *
0.5, time = 400, transition = easing.outExpo } )

        end
        transition.to( event.target, { x = display.contentWidth +
event.target.contentWidth * 0.5, time = 400, transition = easing.outQuad } )
    end

    -- Back button
    local backButton = widget.newButton
    {
        width = 198,
        height = 59,
        label = "Back",
        onRelease = goBack,
    }
    backButton.x = display.contentWidth + backButton.contentWidth * 0.5
    backButton.y = selectedItemAnchor.y + selectedItemAnchor.contentHeight +
backButton.contentHeight
    group:insert( backButton )


    -- Handle row rendering
    local function onRowRender( event )
        local phase = event.phase
        local row = event.row
        local text = "Call log page 1. All pages not shown in ex."
        if not row.isCategory then
            text = row.params.sid
        end

        local rowTitle = display.newText( row, text, 0, 0, nil, 14 )
        rowTitle.x = row.x - ( row.contentWidth * 0.5 ) + ( rowTitle.contentWidth *
0.5 ) + LEFT_PADDING
        rowTitle.y = row.contentHeight * 0.5
        rowTitle:setTextColor( 0, 0, 0 )

    end
```

```
    -- Handle touches on the row
    local function onRowTouch( event )
        local phase = event.phase
        local row = event.target

        if "release" == phase then
            transition.to( tableView, { x = - tableView.contentWidth, time = 400,
transition = easing.outExpo } )
            transition.to( backButton, { x = display.contentCenterX, time = 400,
transition = easing.outQuad } )
            for k, v in pairs(itemSelected) do
                local item = v
                --Update the item selected text
                item.text = k.." "..row.params[rowDisplayData[k]]

                --Transition out the list, transition in the item selected text and
the back button

                transition.to( item, { x = display.contentCenterX, time = 400,
transition = easing.outExpo } )

            end
        end
    end

    -- Create a tableView
    tableView = widget.newTableView
    {
        top = 32,
        width = 320,
        height = 400,
        --listener = tableViewListener,
        onRowRender = onRowRender,
        --onRowUpdate = onRowUpdate,
        onRowTouch = onRowTouch,
    }
    group:insert( tableView )


    -- Insert the row into the tableView
    tableView:insertRow
    {
        isCategory = true,
        rowHeight = 35,
```



```
        rowColor ={default = { 150, 160, 180, 200 }, over = { 30, 144, 255 }, },
        lineColor = { 220, 220, 220 }
    }

    -- Create row per call log
    for i, call in ipairs(twilio_response.calls) do
        local rowColor =
        {
            default = { 255, 255, 255 },
            over = { 30, 144, 255 },
        }

        -- Insert the row into the tableView
        tableView:insertRow
        {
            isCategory = false,
            rowHeight = 40,
            rowColor = rowColor,
            lineColor = { 220, 220, 220 },
            params = call
        }
    end

end

function scene:enterScene(event)
    isExiting = false
    tableView = nil
    local group = self.view

    -- Create a spinner widget
    local spinner = widget.newSpinner
    {
        left = 150,
        top = 200,
    }
    group:insert( spinner )

    -- Start the spinner animating
    spinner:start()
    spinner.isVisible = true

    local statusText = display.newText( "Retriving Twilio call logs", 60, 240,
native.systemFont, 20 )
    --statusText.x = 10
```

```lua
        --statusText.y = 235
        statusText:setTextColor(0, 0, 0)
        group:insert( statusText )

        local function request_listener(event)
            if event.success and not isExiting then
                group:remove(spinner)
                spinner:removeSelf()
                spinner=nil
                statusText:removeSelf()
                group:remove(statusText)
                self:createTableLog(event.response)
            else
                statusText.text = "Error retriving Twilio call logs"
            end
        end

        local vars = {Type="Calls"} --retrieve all call logs
        R:request(vars, "GET", request_listener)
    end


    -- Our scene
    function scene:exitScene( event )
        isExiting=true
        if tableView then
            tableView:removeSelf()
            self.view:remove(tableView)
            tableView = nil
        end
    end

    scene:addEventListener("enterScene")
    scene:addEventListener("exitScene")

    return scene
```

```
-----------------------------------------------------------------------------
-- calltab.lua
-- by Stewart Bracken  http://stewart.bracken.bz  stew.bracken@gmail.com
-- Call any number with your twilio phone number
-- Lots of code leveraged from WidgetDemo.
-- Known critical bug: text labels and buttons are recreated on every scene
-- enter but never destroyed, adding up a lot of widgets and whatnot.
-----------------------------------------------------------------------------

local widget = require( "widget" )
local storyboard = require( "storyboard" )
local scene = storyboard.newScene()
local Util = require("Twilio.Util")

-- create a constant for the left spacing of the row content
local LEFT_PADDING = 10
local textMode = false

local toTextField = nil
local fromTextField = nil

local tHeight = 30
local tLeft = 80
local tWidth = 200
local tTop = 80

local function createTextFields(self)
    local function fieldHandler( event )
        if ( "began" == event.phase ) then
            -- This is the "keyboard has appeared" event
            -- In some cases you may want to adjust the interface when the keyboard
               appears.

            -- Show Dismiss Keyboard button if in portrait mode
            if isPortrait then
                clrKbButton.isVisible = true
            end

            textMode = true

        elseif ( "ended" == event.phase ) then
            -- This event is called when the user stops editing a field: for
               example, when they touch a different field

        elseif ( "submitted" == event.phase ) then
```



```
            -- This event occurs when the user presses the "return" key (if
               available) on the onscreen keyboard

            -- Hide keyboard
            native.setKeyboardFocus( nil )
            textMode = false
            clrKbButton.isVisible = false       -- Hide the Dismiss KB button
        end
    end

    toTextField = native.newTextField(tLeft,tTop,tWidth, tHeight)
    toTextField.inputType = "number"
    toTextField.text = CALLER_TO
    toTextField:addEventListener("userInput",fieldHandler)
    self.view:insert(toTextField)

    fromTextField = native.newTextField(tLeft,tTop+tHeight+10,tWidth, tHeight)
    fromTextField.inputType = "number"
    fromTextField.text = CALLER_FROM
    fromTextField:addEventListener("userInput",fieldHandler)
    self.view:insert( fromTextField )
end

function scene:enterScene(e)
    if fromTextField == nil then
        createTextFields(self)
    end
end

function scene:exitScene(e)
    fromTextField:removeSelf()
    toTextField:removeSelf()
    fromTextField=nil
    toTextField=nil
end

function scene:createScene( event )

    local group = self.view

    -- Display a background
    local background = display.newImage( "assets/background.png", true )
    group:insert( background )

    -- Status text box
```

```
        local statusBox = display.newRect( 70, 290, 210, 120 )
        statusBox:setFillColor( 0, 0, 0 )
        statusBox.alpha = 0.4
        group:insert( statusBox )

        -- Status text
        local statusText = display.newText( "Enter Twilio validated numbers to begin",
    80, 300, 200, 0, native.systemFont, 20 )
        statusText.x = statusBox.x
        statusText.y = statusBox.y - ( statusBox.contentHeight * 0.5 ) + (
    statusText.contentHeight * 0.5 )
        group:insert( statusText )

        ---------------------------------------------------------------------------------
             -------------
        -- widget.newSpinner()
        ---------------------------------------------------------------------------------
             -------------

        -- Create a spinner widget
        local spinner = widget.newSpinner
        {
            left = 274,
            top = 55,
        }
        group:insert( spinner )
        spinner.isVisible = false

        createTextFields(self)

        local toTextFieldLabel = display.newText( "To:", LEFT_PADDING, 80,
    native.systemFont, 16 )
        toTextFieldLabel:setTextColor( 0 )
        group:insert( toTextFieldLabel )

        local fromTextFieldLabel = display.newText( "From:", LEFT_PADDING,
    tTop+tHeight+10, native.systemFont, 16 )
        fromTextFieldLabel:setTextColor( 0 )
        group:insert( fromTextFieldLabel )

        local function makeCall( event )
            spinner:start()
            spinner.isVisible = true
            statusText.text = "Requesting call..."
```



```
            local function request_listener(e)
                spinner:stop()
                spinner.isVisible=false
                if e.success then
                    statusText.text = "Call success!"
                else
                    status.text = e.message
                end
                print(Util.to_string(e.response))
            end

            local vars = {Type="Calls", To=toTextField.text, From = fromTextField.text,
    Url="http://demo.twilio.com/docs/voice.xml" }
            R:request(vars, "POST", request_listener)
        end

        local callButton = widget.newButton
        {
            left = tLeft,
            top = tTop+tHeight*2+10,
            width = tWidth,
            height = tHeight,
            id = "callButton",
            label = "CALL",
            onRelease = makeCall,
        }
        group:insert( callButton )

    end

    scene:addEventListener( "createScene" )
    scene:addEventListener( "enterScene" )
    scene:addEventListener( "exitScene" )

    return scene
```

```
if ( string.sub( system.getInfo("model"), 1, 4 ) == "iPad" ) then
   application =
   {
      content =
      {
         width = 360,
         height = 480,
         scale = "letterBox",
         xAlign = "center",
         yAlign = "center",
         imageSuffix =
         {
            ["@2x"] = 1.5,
            ["@4x"] = 3.0,
         },
      },
   }
elseif ( string.sub( system.getInfo("model"), 1, 2 ) == "iP" and
display.pixelHeight > 960 ) then
   application =
   {
      content =
      {
         width = 320,
         height = 568,
         scale = "letterBox",
         xAlign = "center",
         yAlign = "center",
         imageSuffix =
         {
            ["@2x"] = 1.5,
            ["@4x"] = 3.0,
         },
      },
   }
   elseif ( string.sub( system.getInfo("model"), 1, 2 ) == "iP" ) then
      application =
      {
         content =
         {
            width = 320,
            height = 480,
            scale = "letterBox",
            xAlign = "center",
            yAlign = "center",
```



```
            imageSuffix =
            {
               ["@2x"] = 1.5,
               ["@4x"] = 3.0,
            },
         },
      }
      elseif ( display.pixelHeight / display.pixelWidth > 1.72 ) then
         application =
         {
            content =
            {
               width = 320,
               height = 570,
               scale = "letterBox",
               xAlign = "center",
               yAlign = "center",
               imageSuffix =
               {
                  ["@2x"] = 1.5,
                  ["@4x"] = 3.0,
               },
            },
         }

      else
         application =
         {
            content =
            {
               width = 320,
               height = 512,
               scale = "letterBox",
               xAlign = "center",
               yAlign = "center",
               imageSuffix =
               {
                  ["@2x"] = 1.5,
                  ["@4x"] = 3.0,
               },
            },
         }

      end
```

```lua
--loginwindow.lua
--Login window works but I haven't completed it, ignore this file.

local widget = require( "widget" )
local storyboard = require( "storyboard" )
local scene = storyboard.newScene()


local TwilioRestClient = require "Twilio.TwilioRestClient"
local auth = require "tests.auth" or {}
--Plug in your twilio account credentials here in the XXXX's
local ACCOUNT_SID = auth.ACCOUNT_SID or "XXXXXXXX" -- your Account SID
local ACCOUNT_TOKEN = auth.ACCOUNT_TOKEN or "XXXXXX" --your account token

--Outgoing Caller phone number
local CALLER_TO = auth.CALLER_TO or "+1NNNNNNNNN"
--Incoming Caller Phone Number, previously validated with Twilio
local CALLER_FROM = auth.CALLER_FROM or "+1NNNNNNNNN"

--Lets just make our TwilioRestClient global because it's easier to use across
scenes
R = TwilioRestClient.create(ACCOUNT_SID, ACCOUNT_TOKEN)


local Util = require("Twilio.Util")

-- create a constant for the left spacing of the row content
local LEFT_PADDING = 10
local textMode = false

function scene:exitScene(event)
    --storyboard.purgeScene(self)
end

function scene:destroyScene(e)
    storyboard.purgeScene("loginwindow")
end


function scene:createScene( event )
    local group = self.view

    -- Display a background
    local background = display.newImage( "assets/background.png", true )
    group:insert( background )
```



```lua
    local function fieldHandler( event )
        if ( "began" == event.phase ) then
            -- This is the "keyboard has appeared" event
            -- In some cases you may want to adjust the interface when the keyboard
               appears.

            -- Show Dismiss Keyboard button if in portrait mode
            if isPortrait then
                clrKbButton.isVisible = true
            end

            textMode = true

        elseif ( "ended" == event.phase ) then
            -- This event is called when the user stops editing a field: for
               example, when they touch a different field

        elseif ( "submitted" == event.phase ) then
            -- This event occurs when the user presses the "return" key (if
               available) on the onscreen keyboard

            -- Hide keyboard
            native.setKeyboardFocus( nil )
            textMode = false
            clrKbButton.isVisible = false       -- Hide the Dismiss KB button
        end
    end

    local tHeight = 30
    local tLeft = 80
    local tWidth = 200
    local tTop = 80

    local sidTextField = native.newTextField(tLeft,tTop,tWidth, tHeight)
    --toTextField.inputType = "number"
    sidTextField.text = auth.ACCOUNT_SID
    sidTextField:addEventListener("userInput",fieldHandler)
    local toTextFieldLabel = display.newText( "SID:", LEFT_PADDING, 80,
native.systemFont, 16 )
    sidTextField:setTextColor( 0 )
    group:insert( sidTextField )

    local tokenTextField = native.newTextField(tLeft,tTop+tHeight+10,tWidth,
tHeight)
```

```
        --fromTextField.inputType = "number"
        tokenTextField.text = auth.ACCOUNT_TOKEN
        tokenTextField:addEventListener("userInput",fieldHandler)
        local fromTextFieldLabel = display.newText( "Token:", LEFT_PADDING,
    tTop+tHeight+10, native.systemFont, 16 )
        tokenTextField:setTextColor( 0 )
        group:insert( tokenTextField )



             ---------------------------------------------------------------------------
             ------------------
        -- widget.newSpinner()
        ---------------------------------------------------------------------------
             -------------

        -- Create a spinner widget
        local spinner = widget.newSpinner
        {
            left = 274,
            top = 55,
        }
        group:insert( spinner )

        -- Start the spinner animating
        --spinner:start()
        spinner.isVisible = false


        local function createTabBar()


        end


        local function authenticate( event )
            spinner:start()
            spinner.isVisible = true
            --statusText.text = "Requesting call..."

            local function request_listener(e)
                spinner:stop()
                spinner.isVisible=false
                if e.success then
                    --login
```



```
                    group.isVisible=false
                    createTabBar()
                    storyboard.gotoScene( "calltab" )
                end
            end

            --If twilio accepts get request, credentials are valid and we login
            local vars = {}
            R:request(vars, "GET", request_listener)
        end

        local loginButton = widget.newButton
        {
            left = tLeft,
            top = tTop+tHeight*2+10,
            width = tWidth,
            height = tHeight,
            id = "loginButton",
            label = "Login",
            onRelease = authenticate,
        }
        group:insert( loginButton )

    end

    scene:addEventListener( "createScene" )

    scene:addEventListener( "destroyScene" )
    return scene
```

```
--------------------------------------------------------------------------------
-- main.lua
-- by Stewart Bracken  http://stewart.bracken.bz  stew.bracken@gmail.com
-- Sample Corona app that uses TwilioRestClient
-- Please ignore bugs as this app was not intended to be published.
--------------------------------------------------------------------------------
local mob = require("mobdebug")
if mob then mob.start() end
local TwilioRestClient = require "Twilio.TwilioRestClient"
local auth = require "tests.auth" or {}
--Plug in your twilio account credentials here in the XXXX's
ACCOUNT_SID = auth.ACCOUNT_SID or "XXXXXXXX" -- your Account SID
ACCOUNT_TOKEN = auth.ACCOUNT_TOKEN or "XXXXXX" --your account token
--Outgoing Caller phone number
CALLER_TO = auth.CALLER_TO or "+1NNNNNNNNNN"
--Incoming Caller Phone Number, previously validated with Twilio
CALLER_FROM = auth.CALLER_FROM or "+1NNNNNNNNNN"

--Lets just make our TwilioRestClient global because it's easier to use across
scenes
R = TwilioRestClient.create(ACCOUNT_SID, ACCOUNT_TOKEN)
--Build example GUI
-- Hide the status bar
display.setStatusBar( display.HiddenStatusBar )

--Set background to white
display.setDefault( "background", 255, 255, 255 )

-- Require the widget & storyboard libraries
local widget = require( "widget" )
local storyboard = require( "storyboard" )

-- The gradient used by the title bar
local titleGradient = graphics.newGradient(
    { 189, 203, 220, 255 },
    { 89, 116, 152, 255 }, "down" )

-- Create a title bar
local titleBar = display.newRect( 0, 0, display.contentWidth, 32 )
titleBar.y = titleBar.contentHeight * 0.5
titleBar:setFillColor( titleGradient )

-- Create the title bar text
local titleBarText = display.newText( "Twilio + Corona by Stewart Bracken", 0, 0,
native.systemFontBold, 16 )
```



```
titleBarText.x = titleBar.x
titleBarText.y = titleBar.y

-- Start at calltab
storyboard.gotoScene( "calltab" )

local sceneTransitionOptions = { effect='slideLeft', time="300", }
local tabButtons =
{
    {
        width = 32,
        height = 32,
        defaultFile = "assets/tabIcon.png",
        overFile = "assets/tabIcon-down.png",
        label = "Make a call!",
        onPress = function() storyboard.gotoScene( "calltab",sceneTransitionOptions
); end,
        selected = true
    },
    {
        width = 32,
        height = 32,
        defaultFile = "assets/tabIcon.png",
        overFile = "assets/tabIcon-down.png",
        label = "Send SMS",
        onPress = function() storyboard.gotoScene( "sendsms",sceneTransitionOptions
); end,
    },
    {
        width = 32,
        height = 32,
        defaultFile = "assets/tabIcon.png",
        overFile = "assets/tabIcon-down.png",
        label = "Call logs",
        onPress = function() storyboard.gotoScene( "calllog",sceneTransitionOptions
); end,
    }
}

-- Create a tab-bar and place it at the bottom of the screen
local tabBar = widget.newTabBar
{
    top = display.contentHeight - 50,
    width = display.contentWidth,
    buttons = tabButtons
```

```
    }
```

```lua
-----------------------------------------------------------------------------
-- sendsms.lua
-- by Stewart Bracken  http://stewart.bracken.bz  stew.bracken@gmail.com
-- Send text message with rest api.
-- Lots of code leveraged from WidgetDemo.
-- Known critical bug: text labels and buttons are recreated on every scene
-- enter but never destroyed, adding up a lot of widgets and whatnot.
-----------------------------------------------------------------------------

local widget = require( "widget" )
local storyboard = require( "storyboard" )
local scene = storyboard.newScene()
local Util = require("Twilio.Util")

-- create a constant for the left spacing of the row content
local LEFT_PADDING = 10
local textMode = false

local toTextField = nil
local fromTextField = nil
local bodyTextField = nil

local tHeight = 30
local tLeft = 80
local tWidth = 200
local tTop = 80

local function createTextFields(self)
    local function fieldHandler( event )
        if ( "began" == event.phase ) then
            -- This is the "keyboard has appeared" event
            -- In some cases you may want to adjust the interface when the keyboard
               appears.

            -- Show Dismiss Keyboard button if in portrait mode
            if isPortrait then
                clrKbButton.isVisible = true
            end

            textMode = true

        elseif ( "ended" == event.phase ) then
            -- This event is called when the user stops editing a field: for
               example, when they touch a different field
```



```lua
        elseif ( "submitted" == event.phase ) then
            -- This event occurs when the user presses the "return" key (if
               available) on the onscreen keyboard

            -- Hide keyboard
            native.setKeyboardFocus( nil )
            textMode = false
            clrKbButton.isVisible = false       -- Hide the Dismiss KB button
        end
    end

    toTextField = native.newTextField(tLeft,tTop,tWidth, tHeight)
    toTextField.inputType = "number"
    toTextField.text = CALLER_TO
    toTextField:addEventListener("userInput",fieldHandler)
    self.view:insert(toTextField)

    fromTextField = native.newTextField(tLeft,tTop+tHeight+10,tWidth, tHeight)
    fromTextField.inputType = "number"
    fromTextField.text = CALLER_FROM
    fromTextField:addEventListener("userInput",fieldHandler)
    self.view:insert( fromTextField )

    bodyTextField = native.newTextField(tLeft,tTop+(tHeight+10)*2,tWidth, tHeight*2)
    --bodyTextField.inputType = "number"
    bodyTextField.text = "Your message goes here"
    bodyTextField:addEventListener("userInput",fieldHandler)
    self.view:insert( bodyTextField )
end

function scene:enterScene(e)
    if fromTextField == nil then
        createTextFields(self)
    end
end

function scene:exitScene(e)
    fromTextField:removeSelf()
    toTextField:removeSelf()
    bodyTextField:removeSelf()
    fromTextField=nil
    toTextField=nil
    bodyTextField=nil
end
```

```lua
function scene:createScene( event )

    local group = self.view

    -- Display a background
    local background = display.newImage( "assets/background.png", true )
    group:insert( background )

    -- Status text box
    local statusBox = display.newRect( 70, 290, 210, 120 )
    statusBox:setFillColor( 0, 0, 0 )
    statusBox.alpha = 0.4
    group:insert( statusBox )

    -- Status text
    local statusText = display.newText( "Enter Twilio validated numbers to begin",
80, 300, 200, 0, native.systemFont, 20 )
    statusText.x = statusBox.x
    statusText.y = statusBox.y - ( statusBox.contentHeight * 0.5 ) + (
statusText.contentHeight * 0.5 )
    group:insert( statusText )

    local spinner = widget.newSpinner
    {
        left = 274,
        top = 55,
    }
    group:insert( spinner )
    spinner.isVisible = false

    createTextFields(self)

    local toTextFieldLabel = display.newText( "To:", LEFT_PADDING, 80,
native.systemFont, 16 )
    toTextFieldLabel:setTextColor( 0 )
    group:insert( toTextFieldLabel )

    local fromTextFieldLabel = display.newText( "From:", LEFT_PADDING,
tTop+tHeight+10, native.systemFont, 16 )
    fromTextFieldLabel:setTextColor( 0 )
    group:insert( fromTextFieldLabel )

    local smsTextFieldLabel = display.newText( "Message:", LEFT_PADDING,
tTop+(tHeight+10)*2, native.systemFont, 16 )
    smsTextFieldLabel:setTextColor( 0 )
```



```lua
    group:insert( smsTextFieldLabel )

    local function sendSMS( event )
        spinner:start()
        spinner.isVisible = true
        statusText.text = "Sending sms..."

        local function request_listener(e)
            spinner:stop()
            spinner.isVisible=false
            if e.success then
                statusText.text = "SMS success!"
            else
                status.text = e.message
            end
            print(Util.to_string(e.response))
        end

        --Send a message
        local vars = {Type="Messages", From=fromTextField.text, To =
toTextField.text, Body=bodyTextField.text}
        R:request(vars,"POST", request_listener)
    end

    local sendButton = widget.newButton
    {
        left = tLeft,
        top = tTop+(tHeight+10)*4,
        width = tWidth,
        height = tHeight,
        id = "sendButton",
        label = "Send SMS",
        onRelease = sendSMS,
    }
    group:insert( sendButton )

end
scene:addEventListener( "createScene" )
scene:addEventListener( "enterScene" )
scene:addEventListener( "exitScene" )

return scene
```

```lua
-- test for TwilioRestClient

local Rest = require "Twilio.TwilioRestClient"
--My own authentication credentials are not commited to repository. Get your own!
local auth = require "tests.auth" or {}
local json = require("json")
local Util = require "Twilio.Util"

--Plug in your twilio account credentials here in the XXXX's
local ACCOUNT_SID = auth.ACCOUNT_SID or "XXXXXXXX" -- your Account SID
local ACCOUNT_TOKEN = auth.ACCOUNT_TOKEN or "XXXXXX" --your account token

--Outgoing Caller phone number
local CALLER_TO = auth.CALLER_TO or "+1NNNNNNNNN"
--Incoming Caller Phone Number, previously validated with Twilio
local CALLER_FROM = auth.CALLER_FROM or "+1NNNNNNNNN"

local r = Rest.create(ACCOUNT_SID, ACCOUNT_TOKEN)

local vars={}

local function network_callback(event)
    if ( not event.success ) then
        print( "Unsucessful Response: ", event.message )
    else
        --Now we have a Lua table of the json resonse!
        local response_table = event.response
        print(Util.to_string(response_table))
    end
end

--Make a call:
vars = {Type = "Calls", To = CALLER_TO, From = CALLER_FROM,
Url="http://demo.twilio.com/docs/voice.xml"}
--r:request(vars, "POST", network_callback)

--Make an invalid call, but valid http request
vars.From = nil
r:request(vars, "POST", network_callback)

--Send a message
vars = {Type="Messages", From=CALLER_FROM, To = CALLER_TO, Body="You are looking
sooo good today!"}
--r:request(vars,"POST", network_callback)
```



```lua
--Call Instance Resource
vars = {Type="Calls", InstanceSid = "1234567890"}
--r:request(vars,"GET",network_callback) --Fails unless InstanceSid is correct

--GET calls list
vars = {Type="Calls"}
--r:request(vars,"GET",network_callback) --this will fail
```

```lua
--------------------------------------------------------------------------------
-- TwilioRestClient.lua
-- by Stewart Bracken  http://stewart.bracken.bz  stew.bracken@gmail.com
-- Access the Twilio Rest API within a Corona aplication.
-- License: just friggin use it. Shoot me an email for any questions.
--------------------------------------------------------------------------------

local mime = require "mime"
local json = require("json")
local Util = require "Twilio.Util"

local TwilioRestClient = setmetatable({}, nil)
TwilioRestClient.__index = TwilioRestClient

--------------------------------------------------------------------------------
-- create() - public
-- @param acc_sid - type(string) - Your account SID.
-- @param acc_token - type(string) - Your account token.
-- @param [base_url] - type(string) - Optionally define base Twilio URL
-- @param api_version - type(string) - Optionally define API version.
-- @return - Your new TwilioRestClient instance.
--------------------------------------------------------------------------------
function TwilioRestClient.create(acc_sid, acc_token, base_url, api_version)
   assert(acc_sid and type(acc_sid) == "string",
          acc_token and type(acc_token) == "string",
          base_url == nil or type(base_url) == "string",
          api_version == nil or type(api_version) == "string",
       "Rest.create(acc_sid string, acc_token string[, "..
           "base_url string[, api_version string]]")

   base_url = base_url or "https://api.twilio.com"
   api_version = api_version or "2010-04-01"

   local instance = setmetatable({}, TwilioRestClient)
   instance.sid = acc_sid
   instance.token = acc_token
   instance.base = base_url
   instance.api_version = api_version

   return instance
end

--------------------------------------------------------------------------------
-- buildURL() - private
-- @param client - type(TwilioRestClient) - instance of your Twilio client.
```



```lua
-- @param arg - type(string) - any additional arguments will append to the
--              request URL.
-- @return - type(string) - Twilio http request URL, which in only valid if
--                          all args are Twilio valid.
--------------------------------------------------------------------------------
local function buildURL(client, ...)
    local out = client.base .. "/" .. client.api_version .. "/Accounts/" ..
client.sid
    for i,v in ipairs(arg) do
        out = out .. "/" .. tostring(v)
    end
    out = out .. ".json"
    return out
end


--------------------------------------------------------------------------------
-- formatPhoneNumber() - private
-- @param number - type(string) - phone number in Twilio format 11234567890
-- @return - type(string) - number with + in front if it doesn't have it.
--------------------------------------------------------------------------------
local function formatPhoneNumber(number)
    -- use '%2B' for +
    local s,e = string.find(number,"+")
    if not s then
        number = "+" .. number
    end
    return number
end



--------------------------------------------------------------------------------
-- buildURI() - private
-- @param vars - type(table) - table containing body uri data. For example,
--                             passing {Url=blah.com, To=123456789} returns
--                             'Url=blah.com&To=123456789'
-- @return - type(string)
--------------------------------------------------------------------------------
--Vars is a table containing body uri data ie passing
--{Url=blah.com, To=123456789} returns Url=blah.com&To=123456789
-- Used by post function
local function buildURI(vars)
    local out = ""
    local first = true
    for k,v in pairs(vars) do
        if first then
```

```
                first = false
            end
            --Prevent tables or other unwanted types
            if Util.typechk(v,"string","number") and
               Util.typechk(k,"string","number") then
                if k == "To" or k == "From" then v = formatPhoneNumber(v) end
                out = out .. (not first and "&" or "") .. tostring(k) .. "=" ..
    tostring(v)
            end
        end
        return out
    end

    ------------------------------------------------------------------------------
    -- network_middle_man() - private
    -- @param e - type(table) - event recieved from Corona network.
    -- @param listener - type(function) - user defined network listener
    --  Used internally by TwilioRestClient to format data from server and handle
    --  errors. Formats Twilio data for you :D
    ------------------------------------------------------------------------------
    local function network_middle_man(e, listener)
        --Convert json response into Lua table. Magic!
        e.response = json.decode(e.response)

        --Combine Twilio response errors (ie bad request by user)
        -- and HTTP errors as a general success status
        -- TODO: is this bad practice?
        e.success = (e.status < 300 and e.status > 199) and not e.isError
                    and not e.response.code

        if not e.success then
            e.message = e.response.message
        end

        --Send formatted event data to user defined listener
        listener(e)
    end



    ------------------------------------------------------------------------------
    -- post() - private
    -- @param vars - type(table) - table containing body uri data. For example,
    --                             passing {Url=blah.com, To=123456789} returns
    --                             'Url=blah.com&To=123456789'
    -- @return - type(string)
```



```
    ------------------------------------------------------------------------------
    local function post(client, vars, listener)
        local t = vars.Type
        vars.Type = nil
        local body = buildURI(vars)
        local url = buildURL(client,t)

        local headers = {}
        headers["Content-Type"] = "application/x-www-form-urlencoded"
        headers["Authorization"] =  "Basic "..mime.b64(client.sid..":"..client.token)

        local params = {}
        params.headers = headers
        params.body = body

        network.request(url, "POST", listener, params)
    end

    ------------------------------------------------------------------------------
    -- get() - private
    -- @param vars - type(table) - table containing body uri data. Get requests can
    --                             optionally have an instance SID in which case no
    --                             no body data is sent to Twilio except the ISid.
    -- @return - type(string)
    ------------------------------------------------------------------------------
    --GET requests can optionally have an instance sid, in which
    --case no body data is sent to twilio
    function get(client, vars, listener)
        local t = vars.Type
        local ISid = vars.InstanceSid
        vars.Type = nil
        vars.InstanceSid = nil
        local url = buildURL(client, t, ISid)
        --If InstanceSid is provided, REST doesn't need any properties
        local body = ISid and "" or buildURI(vars)

        local headers = {}
        headers["Content-Type"] = "application/x-www-form-urlencoded"
        headers["Authorization"] =  "Basic "..mime.b64(client.sid..":"..client.token)

        local params ={headers = headers, body = body}

        network.request(url, "GET", listener, params)
    end
```

```lua
--------------------------------------------------------------------------------
-- Request() - public
-- @param vars - type(table) - [key,val] pairs associated with a Twilio request
--               special key 'Type' defines what kind of request you're making.
-- @param method - type(string) - either "POST" or "GET".
-- @param listener - type(function) - function that accepts the http request
--                   events asyncronously.
-- @return the return value comes to the user defined listener(event) function
--         event.success specifies if the request was valid. Refer to
--         event.message for failure information. event.response is a lua table
--         containing your data returned from Twilio. Refer to
--         https://www.twilio.com/docs/api/rest for Twilio response infos.
--------------------------------------------------------------------------------
function TwilioRestClient:request(vars, method, listener)
    assert( type(vars) == 'table',
            method == "POST" or method == "GET",
            type(listener) == 'function',
            "TwilioRestClient:request(vars, method)" )
    vars = (vars and Util.DeepCopy(vars)) or {} --deep copy so I can modify in place
    local net_listener = function(e) network_middle_man(e,listener) end
    if method == "POST" then
        return pcall(function() post(self, vars, net_listener) end)
    elseif method == "GET" then
        return pcall(function() get(self, vars, net_listener) end)
    end
end

return TwilioRestClient
```

```lua
--------------------------------------------------------------------------------
-- Util.lua
-- by Stewart Bracken  http://stewart.bracken.bz  stew.bracken@gmail.com
-- Helper functions required for TwilioRestClient.
--------------------------------------------------------------------------------
local Util = setmetatable({}, nil)

--Set to false for less assert checks and better performance
local doTypeCheck = true

--DeepCopy is the only function in this file necessary for TwilioRestClient.
function Util.DeepCopy(object)
    if type(object) ~= "table" then
        return object
    end
    local new_table = {}
    for index, value in pairs(object) do
        new_table[Util.DeepCopy(index)] = Util.DeepCopy(value)
    end
    return setmetatable(new_table, getmetatable(object))
end

--Taken from http://lua-users.org/wiki/TableSerialization
function Util.table_print (tt, indent, done)
  done = done or {}
  indent = indent or 0
  if type(tt) == "table" then
    local sb = {}
    for key, value in pairs (tt) do
      table.insert(sb, string.rep (" ", indent)) -- indent it
      if type (value) == "table" and not done [value] then
        done [value] = true
        table.insert(sb, "{\n");
        table.insert(sb, Util.table_print (value, indent + 2, done))
        table.insert(sb, string.rep (" ", indent)) -- indent it
        table.insert(sb, "}\n");
      elseif "number" == type(key) then
        table.insert(sb, string.format("\"%s\"\n", tostring(value)))
      else
        table.insert(sb, string.format(
            "%s = \"%s\"\n", tostring (key), tostring(value)))
       end
    end
    return table.concat(sb)
  else
```



```lua
    return tt .. "\n"
  end
end

--Also taken from the above URL
function Util.to_string( tbl )
    if  "nil"       == type( tbl ) then
        return tostring(nil)
    elseif  "table" == type( tbl ) then
        return Util.table_print(tbl)
    elseif  "string" == type( tbl ) then
        return tbl
    else
        return tostring(tbl)
    end
end
--Returns true if checkMe is one of the types passed in to arg
-- Ex multiTypeCheck({1,2,3}, "string", "number") == false
local function multiTypeCheck(checkMe, ...)
    for i, v in ipairs(arg) do
        if type(checkMe) == v then
            return true
        end
    end
    return false
end

--For performance, turn off typechecking at top of Util
Util.typechk = doTypeCheck and multiTypeCheck or function(...) return true end


--Unused because Corona network function encodes URLs as far as I know
function Util.url_encode(str)
  if (str) then
    str = string.gsub (str, "\n", "\r\n")
    str = string.gsub (str, "([^%w %-%_%.%~])",
        function (c) return string.format ("%%%02X", string.byte(c)) end)
    str = string.gsub (str, " ", "+")
  end
  return str
end
return Util
```

```lua
local Util = setmetatable({}, nil)

Util.EPSILON = 0.00001

function Util.DeepCopy(object)
    local lookup_table = {}
    local function _copy(object)
        if type(object) ~= "table" then
            return object
        elseif lookup_table[object] then
            return lookup_table[object]
        end
        local new_table = {}
        lookup_table[object] = new_table
        for index, value in pairs(object) do
            new_table[_copy(index)] = _copy(value)
        end
        return setmetatable(new_table, getmetatable(object))
    end
    return _copy(object)
end

-- Merges two tables, with values from table2 taking precedence over values from
   table1
function Util.MergeTables(table1, table2)
    local newTable = {}

    if (table1) then
        for key, value in pairs(table1) do
            newTable[key] = value
        end
    end

    if (table2) then
        for key, value in pairs(table2) do
            newTable[key] = value
        end
    end
    return newTable
end

-- Removes a value from an array
function Util.FindAndRemove(tab, item)
    for i, testItem in ipairs(tab) do
        if (testItem == item) then
```



```lua
            table.remove(tab, i)
            return true
        end
    end

    return false
end

function Util.DegToRad(degrees)
    return degrees * math.pi / 180.0
end

function Util.RadToDeg(radians)
    return radians*180.0/math.pi
end

function Util.DeclareGlobal(name, value)
    rawset(_G, name, value or {})
end

function Util.UndeclareGlobal(name)
    rawset(_G, name, nil)
end

function Util.GlobalDeclared(name)
    return rawget(_G, name) ~= nil
end

local function denyNewIndex(_ENV, var, val)
    error("Attempt to write undeclared object property: \"" .. tostring(var) ..
"\"")
end

local function denyUndefinedIndex(_ENV, var)
    error("Attempt to read undeclared object property: \"" .. tostring(var) .. "\"")
end

function Util.lockObjectProperties(...)
    for _, object in ipairs(arg) do
        local meta = getmetatable(object)
        if (meta) then
            assert(meta.__newindex == nil, "Can't lock object - it already has a
__newindex set")
            meta.__newindex = denyNewIndex
        else
```

```
            meta = {__newindex = denyNewIndex}
        end
    end
end

function Util.unlockObjectProperties(...)
    for _, object in ipairs(arg) do
        local meta = getmetatable(object)
        assert(meta and meta.__newindex == denyNewIndex, "Can't unlock object - it
wasn't locked with lockObjectProperties")
        meta.__newindex = nil
    end
end

function Util.errorOnUndefinedProperty(...)
    for _, object in ipairs(arg) do
        local meta = getmetatable(object)
        if (meta) then
            assert(meta.__index == nil, "Can't set object to error on undefined -
it already has an __index set")
            meta.__index = denyUndefinedIndex
        else
            meta = {__index = denyUndefinedIndex}
        end
    end
end

function Util.printProps(obj, message)
    if (message) then
        print(message)
    end
    for k, v in pairs(obj) do
        print("\t", tostring(k) .. ":", v)
    end
end

function Util.lerp(a, b, t)
    return a + (b - a) * t
end

function Util.sign(a)
    if a < 0 then
        return -1
    else
        return 1
```



```
    end
end

-- override print() function to improve performance when running on device
-- and print out file and line number for each print
local original_print = print
if ( system.getInfo("environment") == "device" ) then
    print("Print now going silent. With Love, util.lua")
   print = function() end
else
    print = function(message)
        local info = debug.getinfo(2)
        local source_file = info.source
        --original_print(source_file)
        local debug_path = source_file:match('%a+.lua')
        if debug_path then
            debug_path = debug_path  ..' ['.. info.currentline ..']'
        end
        original_print(((debug_path and (debug_path..": ")) or "")..message)
    end
end

-------------------------------------------------------------
-- Array Utilities
-------------------------------------------------------------

--Concatenate array table B onto the end of array table A
function Util.arrayConcat(A,B)
    local iA = #A
    for i = 1, #B do
        A[i+iA] = B[i]
    end
    return A
end

--Concats B onto the end of A but doesn't allow duplicates from B in A
-- if endA is set to #A, then we won't be checking for duplicates in B while adding
   to A
function Util.arrayConcatUnique(A,B,endA)
    endA = endA or #A
    local offset = 0
    for i = 1, #B do
        if Util.arrayContains(A,B[i],endA) then offset = offset + 1
        else
            A[i+endA-offset] = B[i]
```

```lua
            end
        end
        return A
    end

    --endA limits index depth we check for duplicates in array A
    --returns index in which the obj was found in the array A
    function Util.arrayContains(A, obj, endA)
        if (endA and endA > #A) or not endA then
            endA = #A
        end
        --pretty sure this does the above, but the above is more clear
        --endA = (endA and endA < #A and endA) or #A
        for i=1, endA do
            if A[i]==obj then return true, i end
        end
        return false, nil
    end

    --Return a new table with all mutually exclusive elements in A & B
    function Util.getUniqueArray(A,B)
        local unique = {}
        local duplicatesInB = {} --indices of diplicates found in B
        for i=1, #A do
            local doesContain, indexB = Util.arrayContains(B, A[i])
            if not doesContain then
                table.insert(unique,A[i]) -- object A[i] is unique to A
            else
                table.insert(duplicatesInB, indexB) --B[indexB] is equal to A[i]
            end
        end
        --Dup check for all non-known duplicates in B
        --All elements {in A and not in B} are in the unique table.
        --Sort it so we can traverse indices linearly
        table.sort(duplicatesInB)
        local prevStartI = 1
        for i = 1, #duplicatesInB do
            local startI, endI = prevStartI, duplicatesInB[i]-1 --end before dup
            for j = startI, endI do
                if not Util.arrayContains(A, B[j]) then
                    table.insert(unique,B[j])
                end
            end
            prevStartI = endI+2 --skip over the duplicate object index
        end
```



```lua
        --Any leftovers after the last known duplicate in B
        if prevStartI <= #B then
            for i=prevStartI, #B do
                if not Util.arrayContains(A, B[i]) then
                    table.insert(unique,B[i])
                end
            end
        end
        return unique
    end

    --returns new table of elements in A that aren't in B
    function Util.arrayNot(A,B)
        local notSet = {}
        for i=1, #A do
            if not Util.arrayContains(B,A[i]) then
                table.insert(notSet, A[i])
            end
        end
        return notSet
    end


    return Util
```

```python
import sys
import sqlite3

db_name = 'produce.db'
if len(sys.argv) > 1 :
    db_name = sys.argv[1]

conn = sqlite3.connect(db_name)
c=conn.cursor()

c.execute('''CREATE TABLE regions (regionid INTEGER PRIMARY KEY, name text
UNIQUE)''')
c.execute('''CREATE TABLE produces (produceid INTEGER PRIMARY KEY, name text
UNIQUE)''')
c.execute('''CREATE TABLE data (produceid INTEGER, regionid integer, start integer,
end integer)''')

conn.commit()
print("finished creating database with 3 tables", db_name)
```

```
import sys
import sqlite3
import re

db_name = 'produce.db'
if len(sys.argv) <= 2 :
    exit("must provide db name followed by a file to parse")
db_name = sys.argv[1]

conn = sqlite3.connect(db_name)
c=conn.cursor()

def request_usr_fix(regionName, data):
    print('ERROR: ', regionName, data)
    exit()
    return None, None

start_dates = {'January':1, 'February':2,
'March':3,'April':4,'May':5,'June':6,'July':7,'August':8,'September':9,'October':10,
'November':11,'December':12, 'Spring':4, 'Summer':7, 'Fall':10,'Winter':1}
end_dates = {'January':1, 'February':2,
'March':3,'April':4,'May':5,'June':6,'July':7,'August':8,'September':9,'October':10,
'November':11,'December':12, 'Spring':7, 'Summer':10, 'Fall':1,'Winter':4}

# valid inputs:
#  'season' and/through 'season'
#  'month' and/through 'month
#  'month'
#  'season'
#  'year-round'
def insert_produce(regionName, data_line):
    if not regionName or not data_line:
        return
    #data_line
    data = data_line.split(',')
    if len(data)!=2: #exta commas in this line
        #data = [re.match(r'^(.*[,])[^,]*$',data_line).group(0), #everything before
first comma
        #        re.match(r'[^,]+$',data_line)] #everything after last comma
        return #throw it out!
    else:
        produce_name = re.sub(r"'","\\'",data[0].strip())
        if data[1] == "\n":
            return
    try:
```



```
            cleaned_date = re.sub('\(.*\)*','',data[1]) #remove comments at end one line
        except IndexError:
            return # No date range Specified
        #If a string like below appears:
        # Parsnips, April and May and again October through December
        #Then add the produce twice into db
        if re.search(r'\b(and again)\b', cleaned_date):
            split_data = cleaned_date.split('and again')
            if len(split_data) != 2 :
                request_usr_fix(regionName, data_line)
                return
            left = ''.join([produce_name, ', ', split_data[0]])
            right = ''.join([produce_name, ', ', split_data[1]])
            insert_produce(regionName, left)
            insert_produce(regionName, right)
            return
        date_range = re.sub(r'\b(through|and|though|into)\b', '-', cleaned_date) #get a
range
        if re.search(r'(year-round)', date_range):
            date_range = 'January-December'

        #Remove extraneous words and misspellings. This gets nasty, but it works!
        date_range = re.sub(r'\b(mid-)', '', date_range, flags=re.IGNORECASE) #remove
these sequence
        date_range = re.sub(r'\b(mis-)', '', date_range, flags=re.IGNORECASE) #remove
these sequence
        date_range = re.sub(r'\b(early)\b', '', date_range, flags=re.IGNORECASE)
        date_range = re.sub(r'\b(late)\b', '', date_range, flags=re.IGNORECASE)
        date_range = re.sub(r'\b(end of)\b', '', date_range, flags=re.IGNORECASE)
        date_range = re.sub(r'\b(harvested in)\b', '', date_range, flags=re.IGNORECASE)
        date_range = re.sub(r'\b(in)\b', '', date_range, flags=re.IGNORECASE)
        date_range = re.sub(r'\b(various)\b', '', date_range, flags=re.IGNORECASE)
        date_range = re.sub(r'\b(Septmeber)\b', 'September', date_range,
flags=re.IGNORECASE) #Septmeber
        date_range = re.sub(r'\b(Septmber)\b', 'September', date_range,
flags=re.IGNORECASE) #Septmeber
        date_range = re.sub(r'\b(Sept)\b', 'September', date_range,
flags=re.IGNORECASE) #Septmeber
        date_range = re.sub(r'\b(Novemeber)\b', 'November', date_range,
flags=re.IGNORECASE)#Novemeber
        date_range = re.sub(r'\b(p\])', '', date_range, flags=re.IGNORECASE)#p]
        date_range = re.sub(r'\b(fresh)', '', date_range, flags=re.IGNORECASE)#fresh
        date_range = re.sub(r'\b(best)', '', date_range, flags=re.IGNORECASE)#BEST
        date_range = re.sub(r'\b(into)', '', date_range, flags=re.IGNORECASE)#BEST
```

```
        #remove whitespace
        date_range = re.sub(r'\s+', '', date_range)

        start_id=0
        end_id = 0
        if re.search(r'(-)', date_range) : #it contains a range like month-month or
    season-season
            date_range = re.sub(r'-+','-',date_range)
            split_data = date_range.split('-')
            start = split_data[0].capitalize()
            end = split_data[1].capitalize()
        else:
            start = date_range.capitalize()
            end = start
        try:
            start_id = start_dates[start]
            end_id = end_dates[end]
        except KeyError:
            print(date_range, start, end)
            request_usr_fix(regionName, data_line)
            return

        #now we can insert it!!
        try:
            #Insert Region
            s = ["INSERT INTO regions(name) VALUES( '", regionName, "')"]
            c.execute(''.join(s))
        except sqlite3.IntegrityError:
            pass #We've already added this region, just skip it.
        except sqlite3.OperationalError:
            print('ERROR',regionName)
        try :
            #Insert Produce
            s = ["INSERT INTO produces(name) VALUES( '", produce_name, "')"]
            c.execute(''.join(s))
        except sqlite3.IntegrityError:
            pass
        except sqlite3.OperationalError:
            print('DING',produce_name)

        c.execute("SELECT produceid FROM produces WHERE produces.name = ?",
    (produce_name,) )
        produce_id = c.fetchone()[0] #returns a tuple with first element the produce id
        c.execute("SELECT regionid FROM regions WHERE regions.name = ?", (regionName,) )
        region_id = c.fetchone()[0] #returns a tuple with first element the region id
```



```
        #try:
        c.execute("INSERT INTO data(produceid, regionid, start, end) VALUES(?,?,?,?)",
    (produce_id, region_id, start_id, end_id) )
        #except sqlite3.InterfaceError:
        #    print("BAD INSERT: ",(produce_id, region_id, start_id, end_id))
            #exit()
        #Insert this produce data
        #s = ["INSERT INTO data VALUES( ", produce_name, "')"]
        #c.execute(''.join(s))

    #region data should be a text file named the region only, no extension
    for i in range(2, len(sys.argv)):
        #print(sys.argv[i])
        f = open( sys.argv[i], "r")
        region = sys.argv[i].split('/')[-1]
        for line in f:
            if len(line) >2 :
                insert_produce(region, line)
        f.close()

    conn.commit()

    #Test
    #c.execute("SELECT regionid, produceid FROM data NATURAL JOIN regions")
    #c.execute("SELECT produceid FROM data, regions WHERE data.regionid =
    regions.regionid")
    #print(c.fetchall())

    print ("done")

    conn.close()
```

Apples, late June through early October (cold storage until spring)
Asparagus, March through June
Basil, May through October
Beets, April through July (year-round from storage)
Blackberrries, late June through early September
Blueberries, late May through early August
Broccoli, late May through early August
Cabbage, late April through early July
Cantaloupes, June through September
Carrots, year-round
Cauliflower, March through June
Chard, October through June
Chicories, fall and winter
Chiles, June through October
Clementines, December
Collard greens, October through June
Corn, late May through August
Cucumbers, late May through early November
Eggplant, late May through early October
Fava beans, February through May
Fennel, October through April
Figs, late July through early October
Garlic, harvested in June (cured and stored year-round)
Grapes, late July through early October
Green beans, late May through early November
Green onions/scallions, January through June
Herbs
Kale, October through June
Leeks, April through August
Lettuce, March through early July
Mandarins, November and December
Melons, late June through September
Mint, year-round
Morels, spring
Mushrooms (cultivated), year-round
Mushrooms (wild), spring through fall
Nectarines, late May through early September
Nettles, March and April
New Potatoes, May
Okra, June through October
Onions, late April through early November (stored year-round)
Oranges, November through January
Oregano, year-round
Parsley, year-round
Parsnips, November through March

Peaches, late May through early September
Pears, August through November
Pea greens, March through May
Peanuts, May through August
Peas and pea pods, late April through early July
Pecans, year-round
Peppers (sweet), June through October
Persimmons, late September through December
Plums & pluots, July and August
Potatoes, late May through August (available from storage year-round)
Pumpkins, late September through early November
Radishes, March through June
Radishes (daikon, watermelon, other large varieties), October through March
Raspberries, June and July
Rhubarb, February through May
Rosemary, year-round
Rutabagas, late September through early December
Sage, year-round
Shallots, June and July (from storage all year)
Shelling beans, July through November
Snap peas/snow peas/pea pods, , late April through early July
Sorrel, year-round
Spinach, late March through early July
Squash (summer), late April through September
Squash (winter), late August through December
Strawberries, late March through early July
Sweet potatoes, harvested July through November but available from storage year-round
Tangerines, December
Thyme, year-round
Tomatoes, June through October
Turnips, January through April
Watermelons, June through September
Winter Squash, late August through December
Zucchini, late April through September
Zucchini Blossoms, late April through September

```
#! /bin/bash
rm produce.db
python create_produce_table.py
python parse_produce.py produce.db regions/*
```

```
import sys
import sqlite3

db_name = 'produce.db'
if len(sys.argv) > 1 :
    db_name = sys.argv[1]

conn = sqlite3.connect(db_name)
c=conn.cursor()

c.execute("SELECT regionid FROM regions WHERE regions.name = 'California'")
cali_id = c.fetchone()[0]
print("CALI:",cali_id)
c.execute("SELECT produceid FROM data WHERE data.regionid = ?", (cali_id, ))
cali_produce = c.fetchall()
print(len(cali_produce),cali_produce)

c.execute("SELECT produceid FROM produces WHERE produces.name = 'Asparagus'")
asparagus_id = c.fetchone()[0]
print("Asparagus: ", asparagus_id)
#x.execute("SELECT

conn.close
```