DCL

*Bee logo copyright of Dreamworks Animation

Our Team

William Essilfie

Project Manager

Craig Rhodes

System Architect

Chang Liu

Testing Master

Ashutosh Nanda

Language Guru

BuzzFeed

PROMOTED BY JUST DANCE 2017 TRENDING 59,758 VIEWS

You Need To See What YouTube Is Doing To "Bee Movie" Right Fucking Now

BEE MOVIE BUT THE *SEINFELD* THEME PLAYS EVERY TIME THEY SAY "BEE."

Posted on November 28, 2016, at 4:03 p.m.

The bee movie trailer but every time they say bee it does the whole trailer really fast

Avoid at All Costs Subscribe 18,620

290,594 views

Add to Share More 7,009 209

Bee movie trailer is played four times and everytime they say "bee" video speeds up

banzai27 Subscribed 4,663

59,900 views

Add to Share More 992 53

The bee movie trailer except every time they say bee Barry likes jazz

Avoid at All Costs Subscribe 19,738

46,141 views

Add to Share More 1,224 8

What's so great about DCL?

File I/O

DCL compiles down to LLVM making it cross-platform

Tilde Operator

Arrays

Global Assignment

Callbacks

Programming in DCL

DCL Syntax

Comments

```
/* DCL Comment */
```

Operators

```
+ - = / ! ~
= > < >= <=
!= &&  || #
```

Arrays

```
string[2] array = ["hello", "world"];
print_line(array{|0|});
/* prints the string "hello" */
```

DCL Syntax (continued)

If/Else

```
if (1) {
    print_line("test was true");
}

else {
    print_line("test was false");
}
```

For/While

```
for(int i=0; i < 10, i = i + 1) {
    print_line(i);
}
```

```
while(i < j) {
    print_line(i);
}
```

File I/O

```
/* opens the file and returns as a string */
string beemovie = read("beemovie.txt");
/* writes beemovie into a new file */
write(beemovie, "beemovie_copy.txt")
```

Callbacks

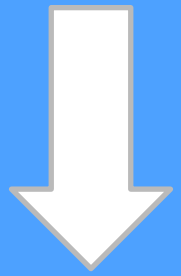
```
string error = "" buteverytime(error != "" && error !=~error) {
    print_line(error);
}
```

buteverytime

Tilde(~) operator

How Callbacks Work

```
string error = "" buteverytime(error != "" && error !=~error) {
    print_line(error);
}
```



```
store { i32, i8* } %actual_arr_literal32, { i32, i8* }* %error
call void @__error({ i32, i8* } %actual_arr_literal32)
%x = alloca i32
store i32 2, i32* %x
call void @__error({ i32, i8* } %actual_arr_literal32)
%x33 = load i32* %x
%print34 = call i32 (i8*, ...)* @printf(i8* getelementptr inbounds ([4 x i8]* @fmt3, i32 0, i32 0), i32 %x33)
call void @__error({ i32, i8* } %actual_arr_literal32)
```

Making DCL

Tools Used

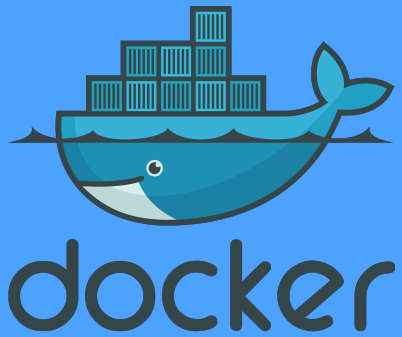
- Facebook Messenger for keeping in contact

- Keeping track of progress and product backlog

- GitHub for version control

Tools Used (continued)

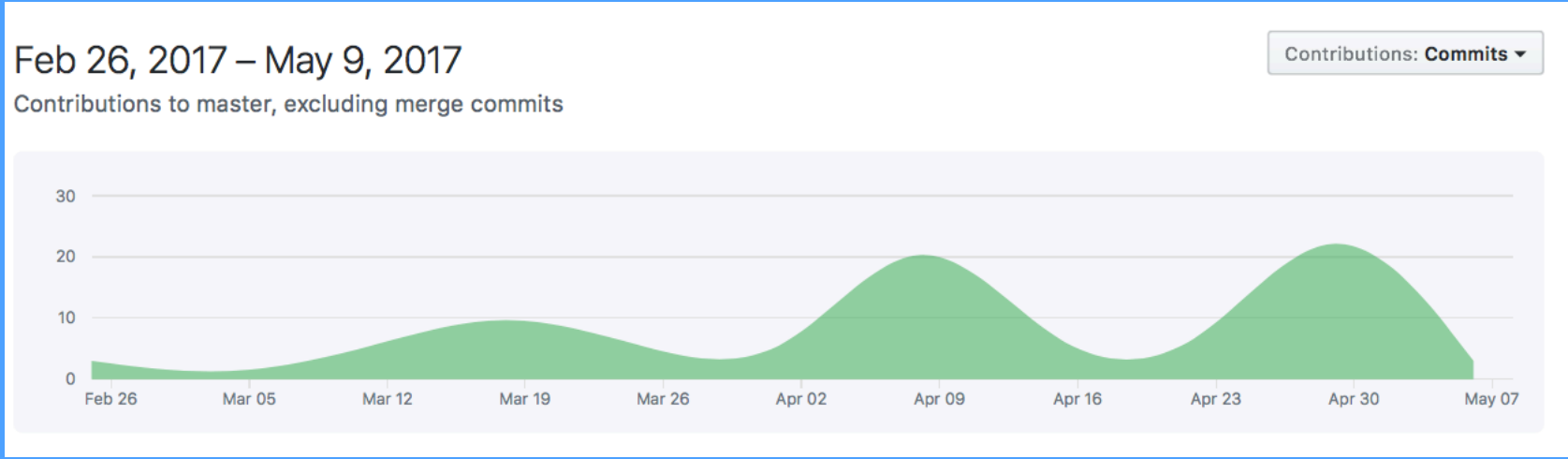
- OCaml for building the language.



- Docker for providing consistent development/ production environment

- LLVM for making language cross-platfom compatible

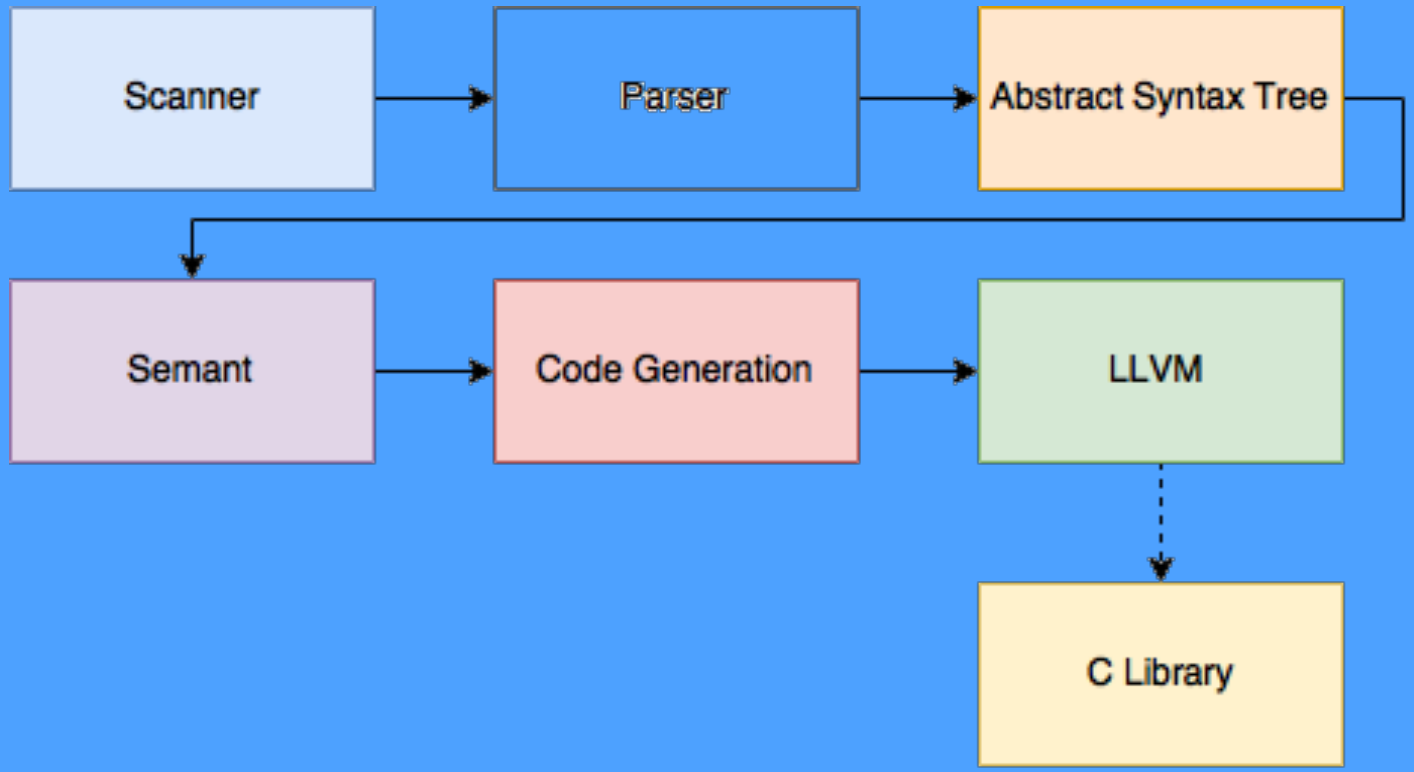
GitHub Commit History



TDD & Continuous Integration

- TDD
- New DCL-specific tests
- Fixing errors as discovered
- Creating tests for bugs found

Architecture

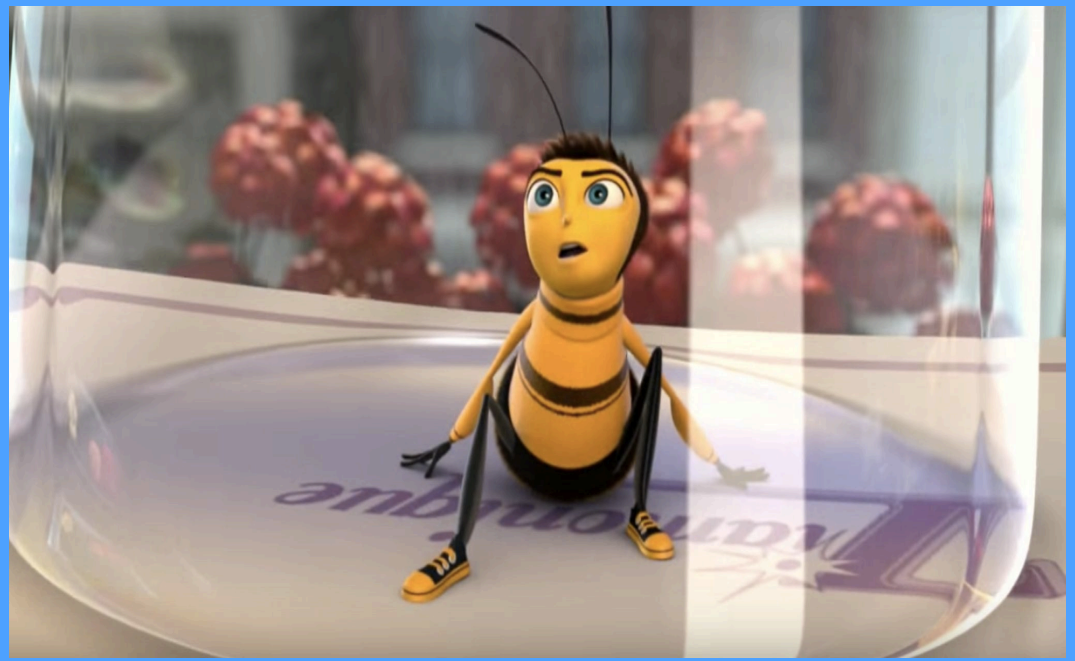


Lessons learned from DCL

- Plan Ahead
- Start early
- In Person Meetings > Online Meetings
- Pair programming
- Communication is key
- Understand MicroC in depth

Demo

"A demo
***buteverytime** you*
say 'bee' it
Becomes 'b + + '"



```
/* Bee Movie but every time they say "bee"... */

string[] split(string whole, string sep) {
    int number_of_parts = 0;
    int looking_for_parts = 0;
    int index;
    for(index = 0; index < #whole; index = index + 1) {
        if(whole{| index |} == sep) {
            if(looking_for_parts) {
                looking_for_parts = 0;
            }
        } else {
            if(!looking_for_parts) {
                number_of_parts = number_of_parts + 1;
                looking_for_parts = 1;
            }
        }
    }
    string[] parts = [number_of_parts of ""];
    int current_index = 0;
    looking_for_parts = 0;
    string current = "";
    for(index = 0; index < #whole; index = index + 1) {
        if(whole{| index |} == sep) {
            if(looking_for_parts) {
                looking_for_parts = 0;
                parts[ current_index ] = current;
                current_index = current_index + 1;
                current = "";
            }
        } else {
            if(!looking_for_parts) {
                looking_for_parts = 1;
            }
            current = current + whole{| index |};
        }
    }
    parts[ current_index ] = current;
    return parts;
}
```

```
string join(string[] parts, string sep) {
    string total = "";
    for(int part = 0; part < #parts; part = part + 1) {
        total = total + parts{| part |} + sep;
    }
    return total;
}

int starts_with(string haystack) {
    return (haystack{| 0 |} == 'b' || haystack{| 0 |} == 'B') &&
           haystack{| 1 |} == 'e' && haystack{| 2 |} == 'e';
}

string current_word = "" buteverytime (starts_with(current_word)) {
    current_word = "     b + +     ";
}

void main() {
    string bee_movie_script = read("bee_movie_script.txt");

    string[] bee_movie_words = split(bee_movie_script, " ");
    string[] modified_bee_movie_words = [#bee_movie_words of ""];

    for(int i = 0; i < #bee_movie_words; i = i + 1) {
        current_word = bee_movie_words{|i|};
        modified_bee_movie_words[i] = current_word;
    }

    write("b++_movie_script.txt", join(modified_bee_movie_words, " "));
}
```

Thanks!

DCL Team