

# Programming Project with Open Source Code (TDDE52/726A89)

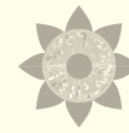
Dániel Varró, August Ernstsson, Máté Földiák

- A note on scale...

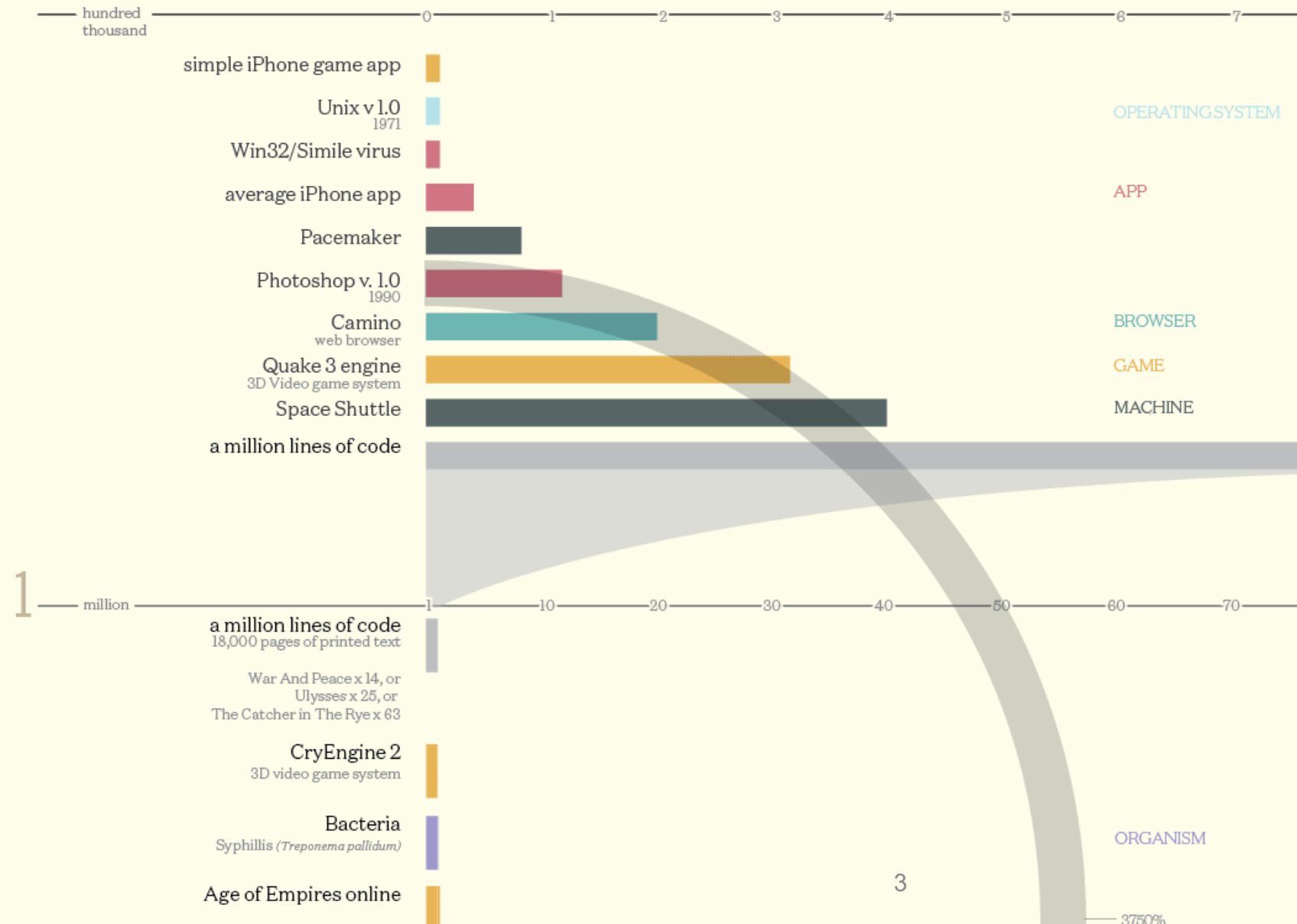
# Codebases

Millions of lines of code

concept & design: David McCandless  
informationisbeautiful.net  
research: Pearl Doughty-White, Miriam Quick  
this graphic is a part of  
knowledge is beautiful [bit.ly/KIB\\_Books](http://bit.ly/KIB_Books)



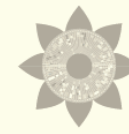
sources NASA, Quora, Ohloh, Wired & press reports  
note some guess work, rumours & estimates  
data [bit.ly/KIB\\_linescode](http://bit.ly/KIB_linescode)



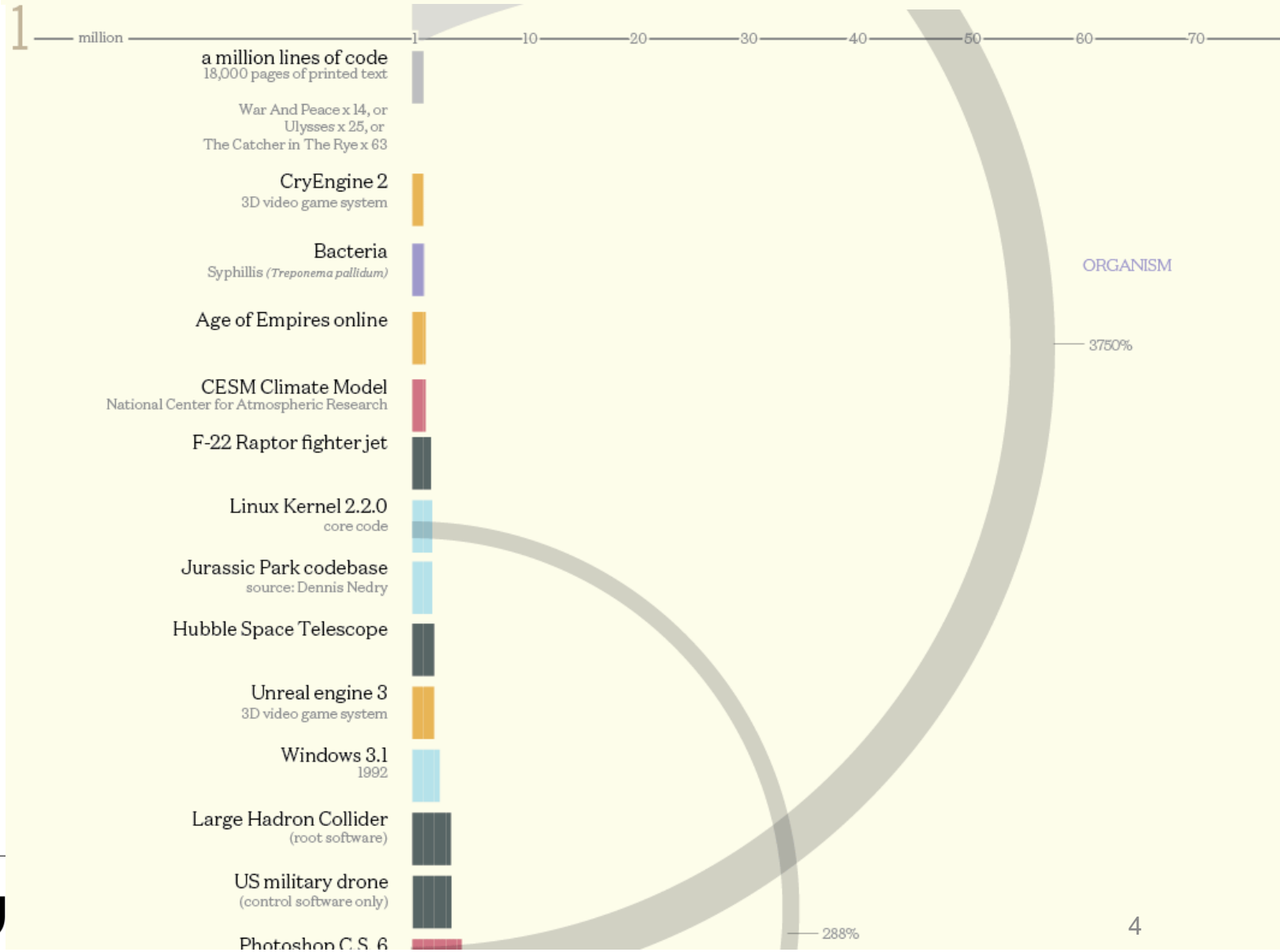
# Codebases

Millions of lines of code

concept & design: David McCandless  
informationisbeautiful.net  
research: Pearl Doughty-White, Miriam Quick  
this graphic is a part of  
knowledge is beautiful [bit.ly/KIB\\_Books](http://bit.ly/KIB_Books)

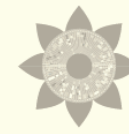


sources NASA, Quora, Ohloh, Wired & press reports  
note some guess work, rumours & estimates  
data [bit.ly/KIB\\_linescode](http://bit.ly/KIB_linescode)



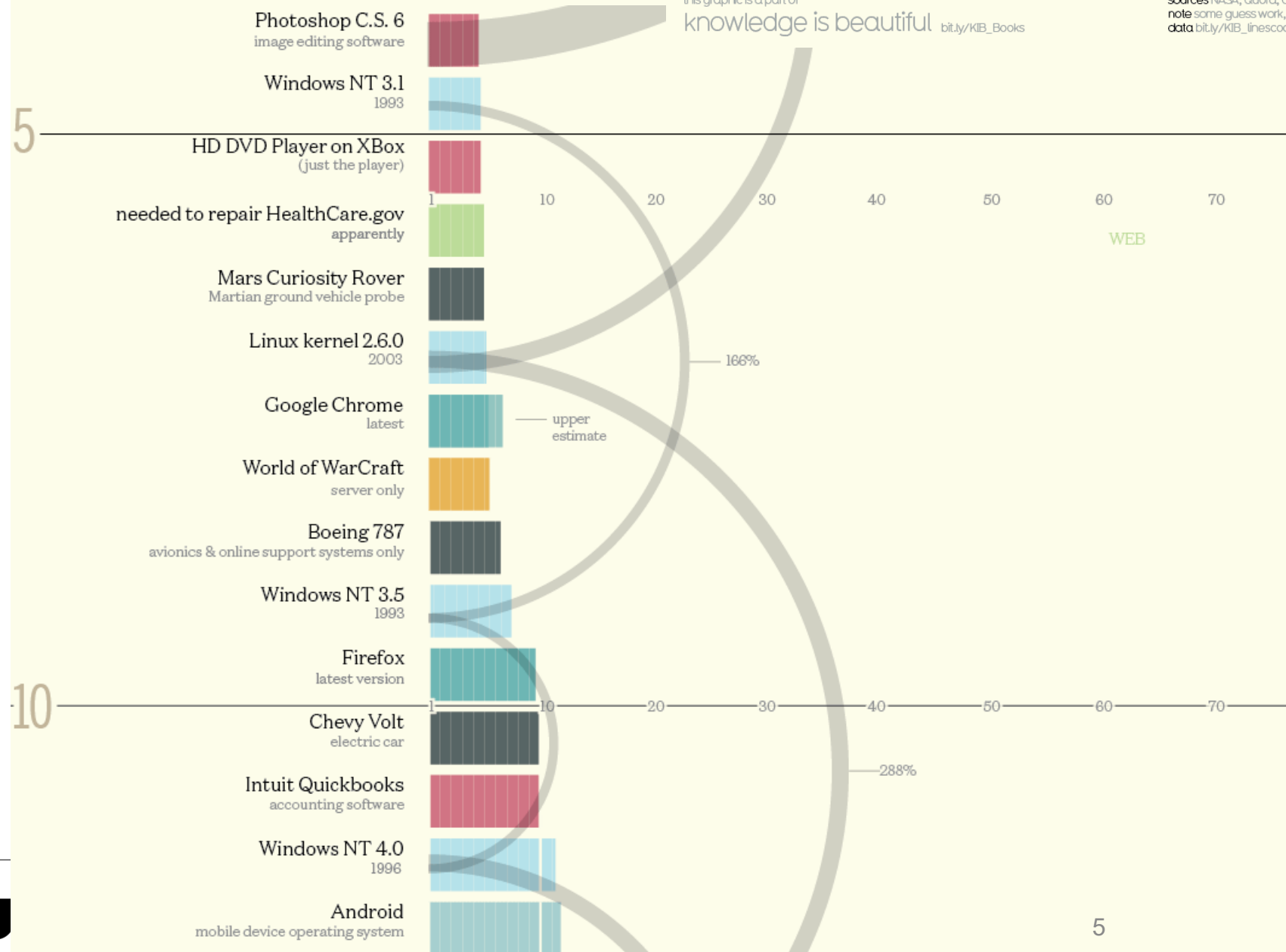
# Codebases

concept & design: David McCandless  
informationisbeautiful.net  
research: Pearl Doughty-White, Miriam Quick



sources NASA, Quora, Ohloh, Wired & press reports  
note some guess work, rumours & estimates  
data [bit.ly/KIB\\_linescode](http://bit.ly/KIB_linescode)

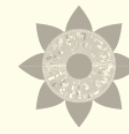
this graphic is a part of  
knowledge is beautiful [bit.ly/KIB\\_Books](http://bit.ly/KIB_Books)



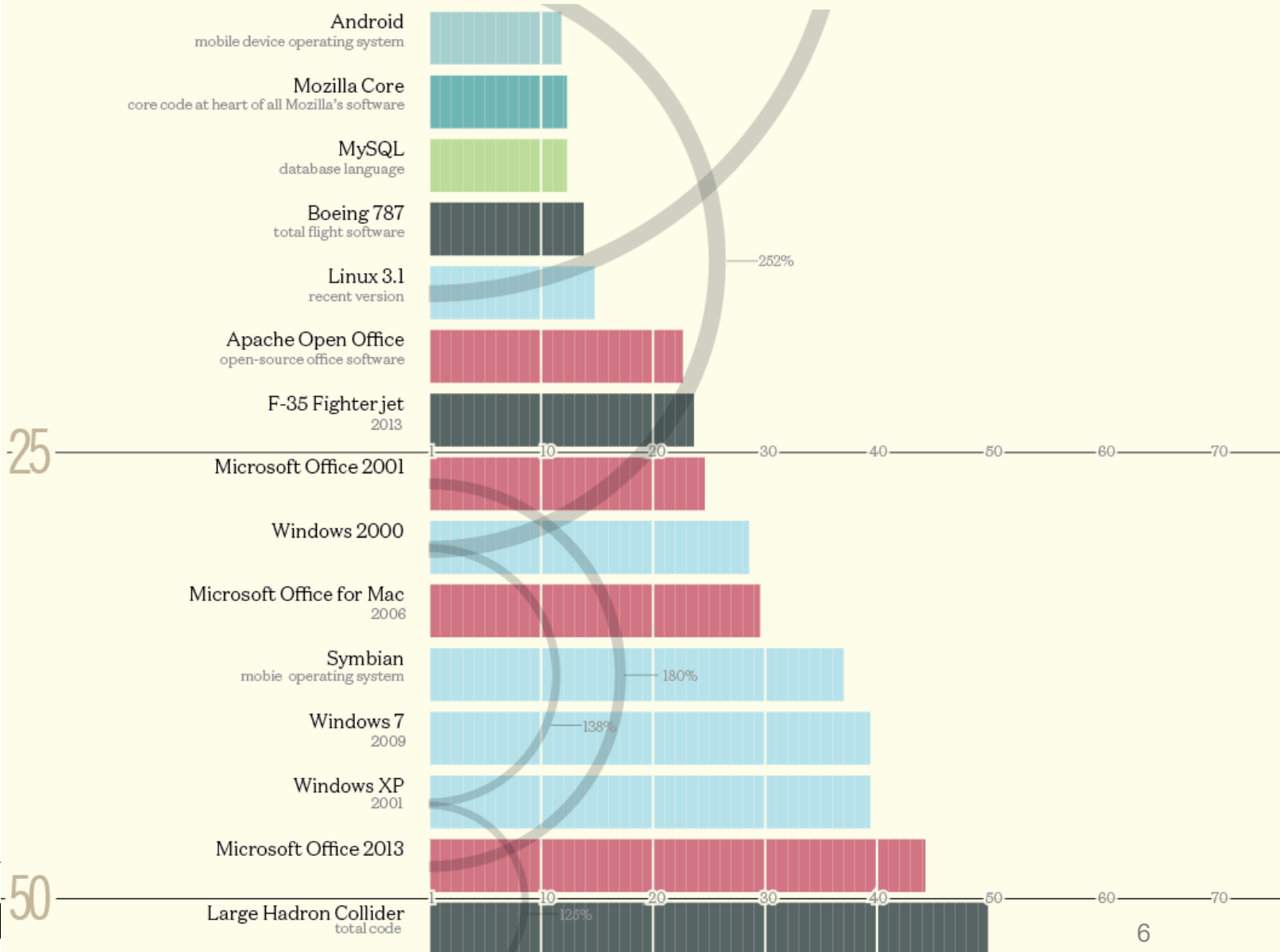
# Codebases

Millions of lines of code

concept & design: David McCandless  
informationisbeautiful.net  
research: Pearl Doughty-White, Miriam Quick  
this graphic is a part of  
knowledge is beautiful [bit.ly/KIB\\_Books](http://bit.ly/KIB_Books)



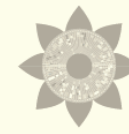
sources NASA, Quora, Ohloh, Wired & press reports  
note some guess work, rumours & estimates  
data [bit.ly/KIB\\_linescode](http://bit.ly/KIB_linescode)



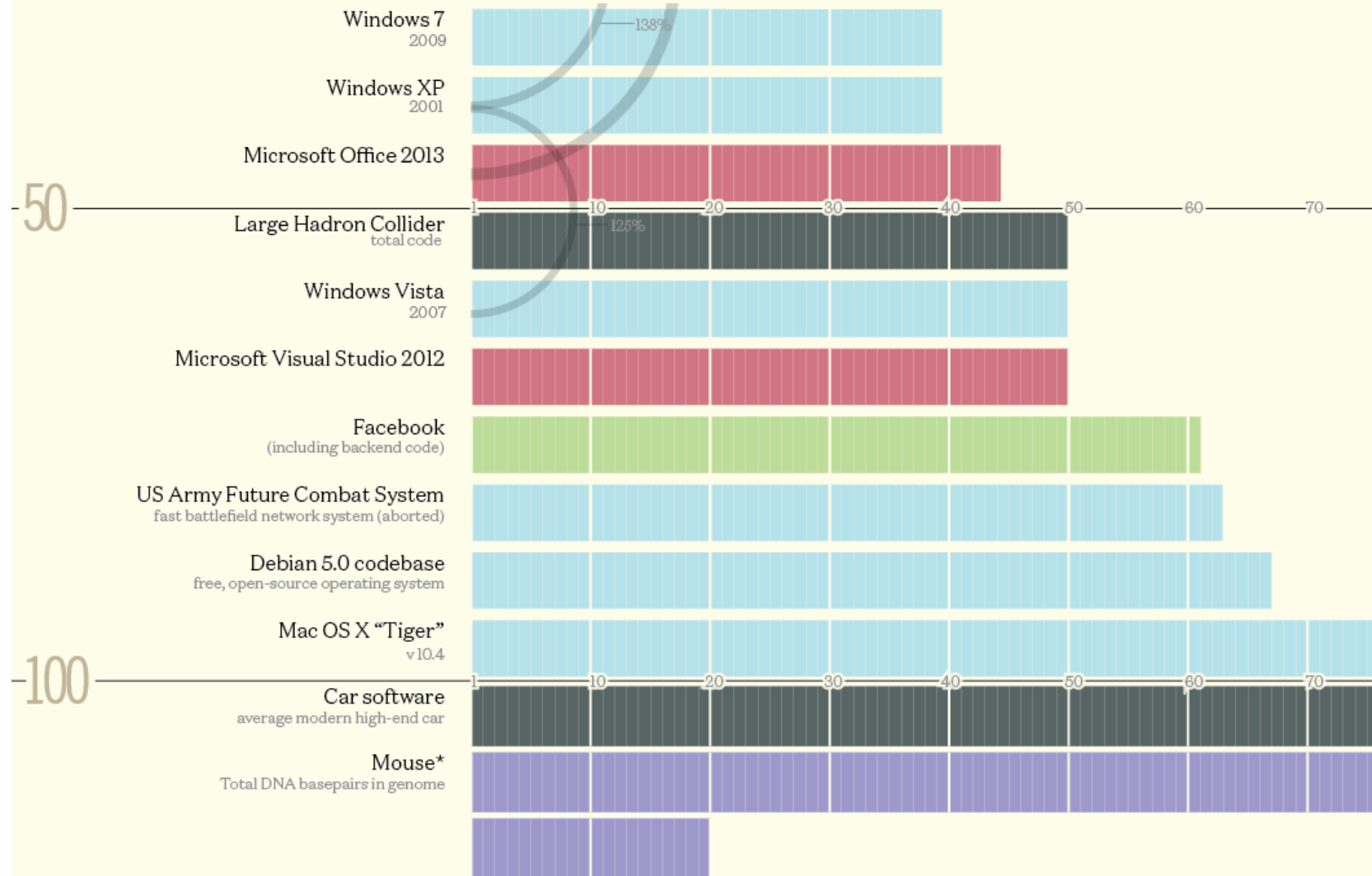
# Codebases

Millions of lines of code

concept & design: David McCandless  
 informationisbeautiful.net  
 research: Pearl Doughty-White, Miriam Quick  
 this graphic is a part of  
 knowledge is beautiful [bit.ly/KIB\\_Books](http://bit.ly/KIB_Books)



sources NASA, Quora, Ohloh, Wired & press reports  
 note some guess work, rumours & estimates  
 data [bit.ly/KIB\\_linescode](http://bit.ly/KIB_linescode)



# Characteristics of ultra-large-scale systems

- Everything is decentralized (data, control,...)
- Requirements: conflicting, diverse, unknown,...
- Continuous evolution
- Heterogeneous, inconsistent and changing elements.
- Eroded people-system boundary
- Failure is the norm

- How can we study this at university?

# Agenda

- Course goals, requirements
- Weekly plan
- Tools and meetings
- Forming teams

# Course goals

1. Use existing conventions and follow established processes to **contribute through software to a distributed, large-scale development project.**
2. Present changes and updates so external parties may approve submissions.
3. Create a time plan and monitor progress through a common development project
4. Use appropriate tools for contemporary, large-scale software development
5. **Independently acquire new knowledge and skills** in order to contribute to a large-scale software project.

# Informal goals

- Portfolio of contributions
- Hands-on experience
- Professional and technical skills
- Creative, hopefully fun and self-steered
- Demonstrating how great LiU students are to the world!

# What happens in practice during the course

- You choose a project to work on.
- You make plans for how to contribute to the project.
- We meet every week to discuss how it has been going, try to support each other and make plans for next week.
- Between meetings you work with the project and use Teams to communicate with the rest of the team.

# Projects

- OSS projects (Github, Gitlab, SourceForge, ...) of sufficient size:
  - 30+ developers,
  - 1000+ commits,
  - actively maintained during the last three months by at least 15 developers
- Project type/difficulty will determine grade possible

# Points table

Level	Technical difficulty	Process difficulty	Development process	Individual contributions
Easy	6	3	4	8
Medium	12	6	8	16
Expert	18	9	12	24

# Grades

- 21-34 points: grade 3
- 35-48 points: grade 4
- 49-63 points: grade 5
  - Example: Lowest technical and process difficulty and highest points on development process and individual contributions:  $6+3+12+24 = 45$ , grade 4
  - Example: Highest technical and process difficulty and lowest points on development process and individual contributions:  $18+9+4+8 = 37$ , grade 4

Technical difficulty	Build and test environment	Domain knowledge requirements	Size
Easy	No configuration needed	No expertise apart from course-level knowledge required	50-400 KLOC, 0-3 external static libraries required.
Medium	File settings and manual build steps	A few specific new algorithms or tools to learn	400 KLOC-1MLOC, several components built and tested against both internal and external components
Hard	Custom configuration languages, build tools	Several weeks or months of study estimated for initial contributions	More than 1 MLOC, several components shipped in multiple versions with dependencies on each other as well as external components.

Process difficulty	Issue tracker	Documentation	Communications
Easy	Contains issues tagged as suitable for beginners	Contains guides and tutorials for new developers	Contains specific communication channels for new developers
Medium	Contains issues marked with difficulty	Contains introduction chapters and suggested readings in documentation	Contains open communication channels
Hard	No metadata on difficulty in issue tracker	No clear introduction to new developers	Restricted communication channels

Development process	Group contributions	Planning and reflection	Engineering practice
Easy	Shares code, tutorials and development tips to other members of the team. Actively participates in project meetings.	Creates a plan for each sprint with sufficient details to guide the work, amounting to a number of hours that correspond the amount of credits given by the course.	Uses the required set of tools and suggested practices for code contributions in the host project
Medium	Shares documentation and material, and also gives valuable feedback on others' contributions	Uses the plan actively and updates the plan to take into account new information and events	Takes care to follow established good practices in continuous development and testing
Hard	Actively helps team members improve, and provides tutorials or other help to team members	Uses external sources from the host project to validate the feasibility of the plan and to update accordingly	Contributes with general quality improvements for the process of developing, building and testing software in the project

<b>Contributions</b>	<b>Quantity of code contributions</b>	<b>Complexity of code contributions</b>	<b>External communications</b>
<b>Easy</b>	At least one contribution that has been accepted by the project	Non-trivial code contribution	Has been able to communicate successfully with at least one developer in the external project on a suggested contribution
<b>Medium</b>	Several contributions that has been accepted by the project	Code contributions that solve different problems or a moderately challenging problem	Has participated in several discussions with project members before and after own contributions
<b>Hard</b>	Several types of contributions (e.g. new features, bug fixes or documentation) that has been accepted by the project	Code contributions that solve a challenging problem	Has successfully proposed new features to the project

# Next steps

- Select a project that you would like to contribute to. You will also form X number of teams, possibly with the same project that you work on.
- Pitch your desired project to others on **September 3rd, Wednesday**
- Write individual project plans, with general ambitions and detailed plans for the first sprint, and additional information before each coming sprint
  - Discussion on preliminary/draft plans on first seminar meeting:  
**September 8th, Monday**
  - Deadline for final plans: **September 12th Friday 23:59**
- 45 minutes weekly / biweekly for discussing progress within each group

# Course evaluation and changes

- No major change in the overall course structure or grading
- We updated the list of recommended projects

# Schedule

Week	Session
36	This lecture
37	First round of project selections
37	Planning report review
38-42	Weekly meetings
45	Mid-semester review
45-49	Bi-weekly meetings
50-51	Final presentations (Dec 8/15)

# Main OS projects selected by students last year

- Oppia
- Zulip
- Godot
- Material-UI
- Mattermost
- Pinta
- *Neovim*
- *Cypress*

# Further experience from last years

- **Zulip** - Welcoming, open but slow communication, but very crowded.
- **Oppia** - Welcoming community, good documentation, but crowded.
- **Godot** - Good but requires some domain knowledge
- **MaterialUI, Mattermost** - Project specific issues (ESLint, Error checks)
- **Pinta** - Difficult setup but great communication
- **Bitwarden** - Challenging. Extensive documentation but very slow communication (2 months) *Not recommended*.
- **Smaller projects** – Generally good experience\*

# Discussion of project proposals

# Next steps

- Select an OSS project and start drafting a plan
- You are free to choose projects as you like, but you will be graded on your ability to help each other out in your student teams as well as your contributions to the project
- Good if people with same/similar project in same group
- Feel free to select a project that is really close to your heart
- Preliminary grouping in next meeting