Hands-On 1:

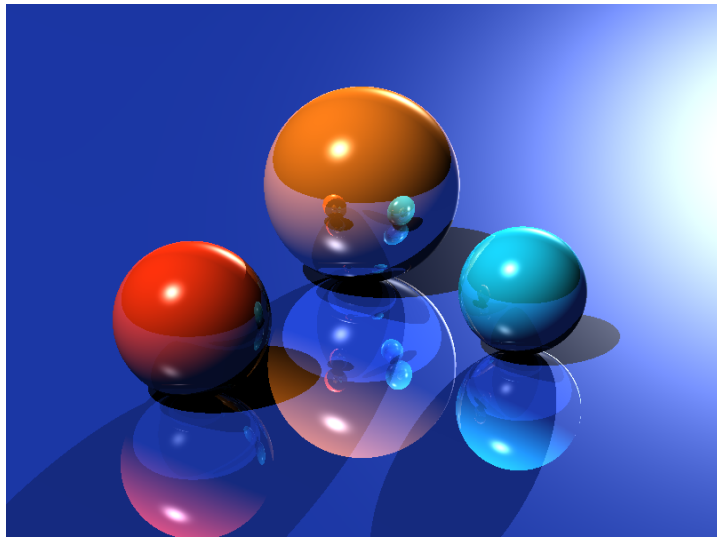
Ray Tracing

(data parallelism)

Safe Harbor Statement

This presentation may contain forward-looking statements that are based on our current expectations. Forward looking statements may include statements about our financial guidance and expected operating results, our opportunities and future potential, our product development and new product introduction plans, our ability to expand and penetrate our addressable markets and other statements that are not historical facts. These statements are only predictions and actual results may materially vary from those projected. Please refer to Cray's documents filed with the SEC from time to time concerning factors that could affect the Company and these forward-looking statements.

Goal: Build a Parallel Ray-Tracer



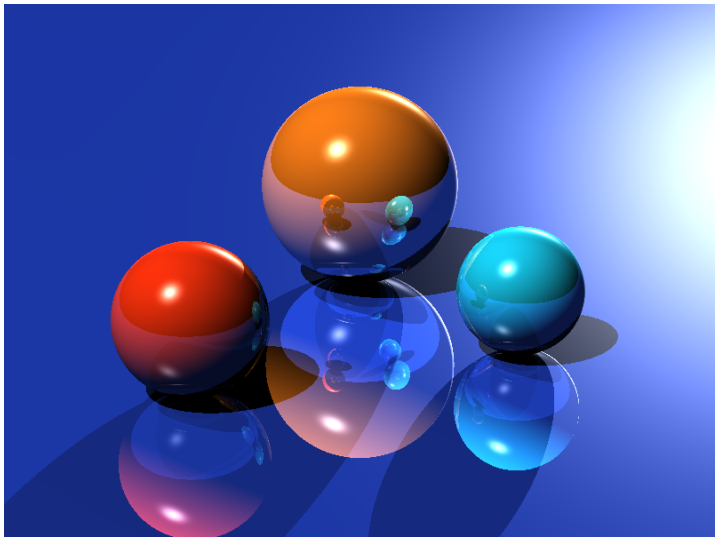
We're providing:

- two scene files
  - **scene:** a simple 4-sphere scene
  - **sphfract:** a fractal pattern of spheres
- framework code: **c-ray.chpl**
  - to read a scene
  - to do the ray tracing computation
  - computePixel() routines:

```
proc computePixel(y: int, x: int): pixelType { … }
proc computePixel(yx: 2*int): pixelType { … }
```

- a helper module to write arrays as PPM or BMP images: **Image.chpl**
- sample output files (*.bmp, *.ppm)

Goal: Build a Parallel Ray-Tracer



Your goal is to:

- declare the array of pixel values
- use computePixel() to fill it
  - serially
  - in parallel
  - optionally: promoted
  - optionally: dynamically load-balanced
  - optionally: distributed
    - may make more sense this afternoon
- compare timings for various versions

(You're also free to pursue any other Chapel coding you like)

- code up a computation of interest to you
- look through primer or benchmark examples in the release
  - $CHPL_HOME/examples
- …

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, and URIKA. The following are trademarks of Cray Inc.: ACE, APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, THREADSTORM. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.