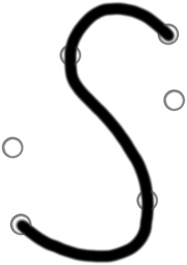


Splines, or: Connecting the Dots

Jens Ogniewski
Information Coding Group

- **Note that not all is covered in the book, especially**
  - Change of Interpolation
  - Centripetal Catmull-Rom and other advanced parameterization schemes
  - Own creation of interpolation methods
  - More advanced Color interpolation

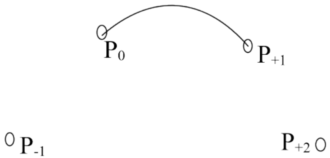
- **These parts will not be part of the exam, but are given merely for your own information**

Interpolation Splines

- **The points are blended together using blending functions**
- **All blending functions are zero or 1 at the control points!**

Catmull-Rom Splines

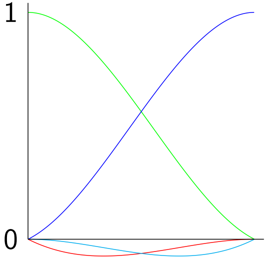
- **Also called cardinal splines**
- **Calculated from 4 control points, defined between the middle two**
- **Specified only by the control points, and a tension parameter to adjust the shape**



Catmull-Rom Splines

$$
\begin{aligned}
\mathbf{F}_{\mathbf{CR}(u)} = & \quad (-\alpha u^3 + 2\alpha u^2 - \alpha u)\mathbf{P_{-1}} \\
+ & \quad ((2-\alpha)u^3 + (\alpha - 3)u^2 + 1)\mathbf{P_0} \\
+ & \quad ((\alpha - 2)u^3 + (3 - 2\alpha)u^2 + \alpha u)\mathbf{P_{+1}} \\
+ & \quad (\alpha u^3 - \alpha u^2)\mathbf{P_{+2}}
\end{aligned}
$$

- **The equations found in the book can be reached by using the “standard” parameterization of** $\alpha = 0.5$



- Red: polynomial for $\mathbf{P_{-1}}$
- Green: polynomial for $\mathbf{P_0}$
- Blue: polynomial for $\mathbf{P_{+1}}$
- Cyan: polynomial for $\mathbf{P_{+2}}$

Catmull-Rom Splines

- **Rewrite:**

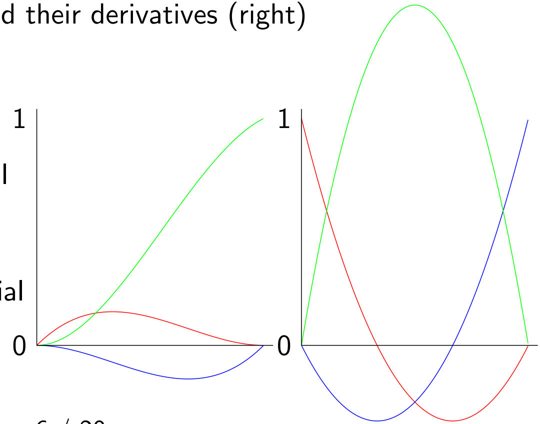
$$
\begin{aligned}
\mathbf{F}_{\mathbf{CR}(u)} = \quad & \alpha u(u-1)^2(\mathbf{P_{+1}} - \mathbf{P_{-1}}) \\
+ \quad & \alpha u^2(u-1)(\mathbf{P_{+2}} - \mathbf{P_0}) \\
+ \quad & u^2(3-2u)(\mathbf{P_{+1}} - \mathbf{P_0}) \\
+ \quad & \mathbf{P_0}
\end{aligned}
$$

- **The first two rows set the tangents in the endpoints $u = 0$ and $u = 1$ respectively, while being 0 at both endpoints**
- **The rest interpolates between $\mathbf{P_0}$ and $\mathbf{P_{+1}}$, while having a zero tangent in both endpoints**
- **If you know what derivatives you want to have in the endpoints, you can set them accordingly (rather than using Catmull-Rom)**

Catmull-Rom Splines

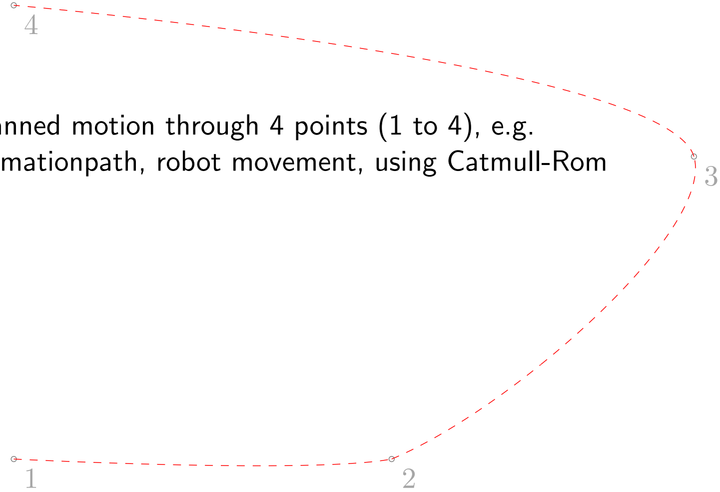
$$
\begin{aligned}
\mathbf{F}_{\mathbf{CR}(u)} = \quad & \alpha u(u-1)^2(\mathbf{P_{+1}} - \mathbf{P_{-1}}) \\
+ \quad & \alpha u^2(u-1)(\mathbf{P_{+2}} - \mathbf{P_0}) \\
+ \quad & u^2(3-2u)(\mathbf{P_{+1}} - \mathbf{P_0}) \\
+ \quad & \mathbf{P_0}
\end{aligned}
$$

- The polynomials (left) and their derivatives (right)
- Red: blending polynomial for the tangent in point $u = 0$
- Blue: blending polynomial for the tangent in point $u = 1$
- Green: blending polynomial for the points between which we interpolate



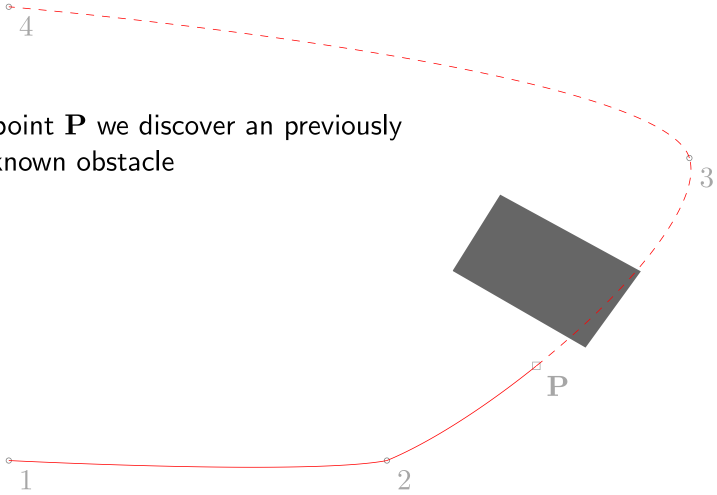
Example: Change of Ongoing Interpolation

- Planned motion through 4 points (1 to 4), e.g. animationpath, robot movement, using Catmull-Rom



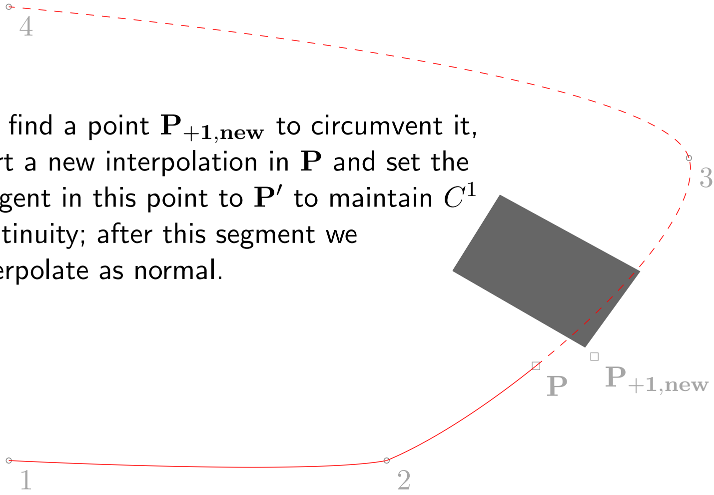
Example: Change of Ongoing Interpolation

- In point $\mathbf{P}$ we discover an previously unknown obstacle



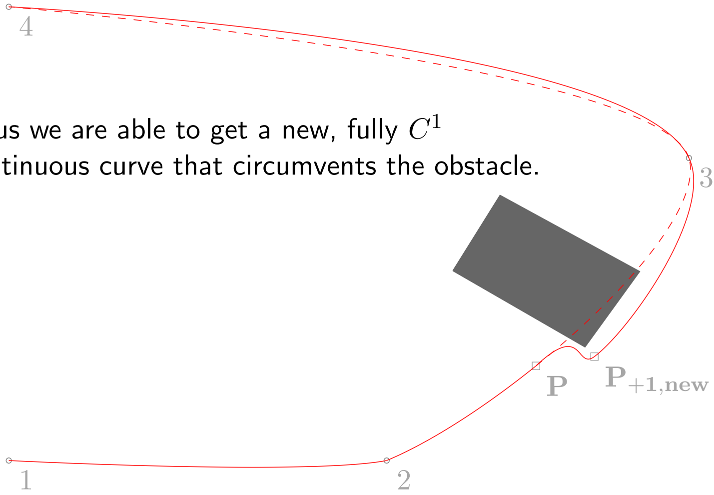
Example: Change of Ongoing Interpolation

- We find a point $\mathbf{P}_{+1,\mathbf{new}}$ to circumvent it, start a new interpolation in $\mathbf{P}$ and set the tangent in this point to $\mathbf{P}'$ to maintain C^1 continuity; after this segment we interpolate as normal.

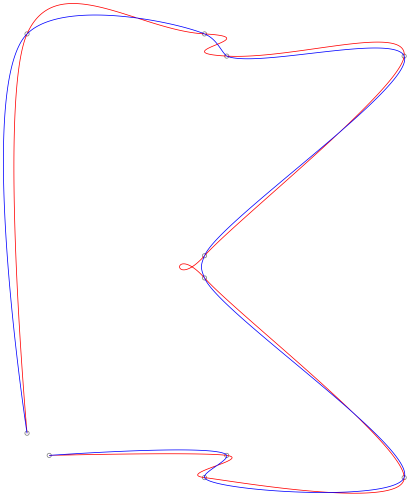


Example: Change of Ongoing Interpolation

- Thus we are able to get a new, fully C^1 continuous curve that circumvents the obstacle.



Parameterization



- Red: “Standard” Catmull-Rom, Blue: Centripetal Catmull-Rom

Parameterization

- **High differences in segment lengths can lead to unwanted artifacts like e.g. loops**
  - Can be solved by normalizing the tangent vectors
  - But needs a similar scaling of the interpolation vector (the vector between the two endpoints of the current vector) as well
  - Correct solution is unfortunately complicated
- **Best solution centripetal Catmull-Rom**
  - Implementations are available
  - For more information:
    Chem Yuksel et al.: "Parameterization and Applications of Catmull-Rom Curves"

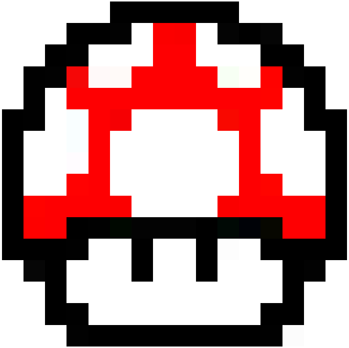
Selecting Your Own Spline

- **Often:**
- **Surfaces, curves: approximating,** G^1 **or** G^2
- **Object animation: interpolating,** C^1
- **Camera movement: interpolating,** G^2
- **Color: trigonometric or interpolating,** C^1

Selecting/Creating Your Own Spline

- **The good: standard implementations available for most algorithms**
- **Also: Interpolation is not that complicated**
  $=>$ with a little effort you can create your own scheme or adapt an existing one
- **The bad: knowing your requirements might be tricky**
- **Often made mistake: overconstraining it, leading to “artifical” look**

Example: Artifact-Free Color-Interpolation



- **In the following we successively refine our requirements**

Example: Artifact-Free Color-Interpolation

- **Linear interpolation: continuation artifacts**
  => needs to be (at least) C^1 continuous

Example: Artifact-Free Color-Interpolation

- **Cubic interpolation: clipping, discoloring artifacts**
  => not allowed to leave bounds (0..255)

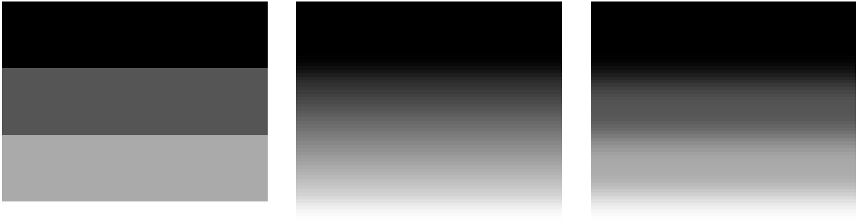
Example: Artifact-Free Color-Interpolation

- **Trigonometric interpolation: ringing artifacts**
  => needs to be bounded by the surrounding colors

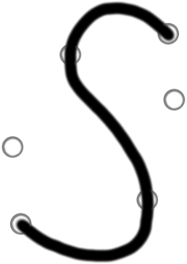
Example: Artifact-Free Color-Interpolation

- **Solution: enforce zero derivatives in the endpoints**

Example: Artifact-Free Color-Interpolation



- **But can lead to "wrong" gradients (see image to the right; input shown to the left)**
- **Solution (middle) a little more involved, for more information see also:**
  Jens Ogniewski "Artifact-free color interpolation"



Thank you for your feedback!

jenso@isy.liu.se