

# Gràfics i Visualització de Dades

## Extensions Visualization Mapping

Anna Puig

# Visualization mapping (extensions de la pràctica)

- Visualization mapping
  - ús de paletes diferents
  - definició de n propietats (localització i gizmo)
  - animacions de les dades
  - mapeig de les dades en una esfera

# Visualization mapping (extensions de la pràctica)

- Ús de paletes diferents

<http://colorbrewer2.org>

The screenshot displays the ColorBrewer 2.0 web application interface. At the top, it says "COLORBREWER 2.0 color advice for cartography". The main interface is divided into several sections:

- Number of data classes:** Set to 3.
- Nature of your data:** Radio buttons for sequential (selected), diverging, and qualitative.
- Pick a color scheme:** Two columns of color swatches labeled "Multi-hue" and "Single hue".
- Only show:** Checkboxes for "colorblind safe", "print friendly", and "photocopy safe".
- Context:** Checkboxes for "roads", "cities", and "borders" (checked).
- Background:** Radio buttons for "solid color" (selected) and "terrain".
- 3-class BuGn:** A section showing three color swatches with their corresponding HEX codes: #e5f5f9, #99d8c9, and #2ca25f.
- Types of Color Schemes:** A modal window explaining three types of color schemes:
  - 1. Sequential schemes:** Suited to ordered data that progress from low to high. Lightness steps dominate the look of these schemes, with light colors for low data values to dark colors for high data values.
  - 2. Diverging schemes:** Put equal emphasis on mid-range critical values and extremes at both ends of the data range. The critical class or break in the middle of the legend is emphasized with light colors and low and high extremes are emphasized with dark colors that have contrasting hues.
  - 3. Qualitative schemes:** Do not imply magnitude differences between legend classes, and hues are used to create the primary visual differences between classes. Qualitative schemes are best suited to representing nominal or categorical data.
- Further reading:** Links to "Brewer, Cynthia A. 1994. Color use guidelines for mapping and visualization. Chapter 7 (pp. 123-147) in Visualization in Modern Cartography" and "Other cartography publications by Cynthia Brewer".
- Legend examples:** Three legends are shown on the right side of the modal window:
  - People per sq. mile:** A sequential legend with four classes: 300.00 to 9316.0 (dark green), 79.6 to 299.9 (medium green), 7.0 to 79.5 (light green), and 1.1 to 6.9 (very light green).
  - Percent of population under 18 by state:** A diverging legend with three classes: 28.0 to 32.2 (dark green), 25.7 to 27.9 (medium green), and 24.0 to 25.6 (light green). A critical value of 25.7 is noted.
  - Race or ethnicity:** A qualitative legend with four classes: Hispanic (yellow), White (orange), Black (purple), and Asian (green).

The background of the application shows a map of the United States with a color scheme applied to the states.

# Visualization mapping (extensions de la pràctica)

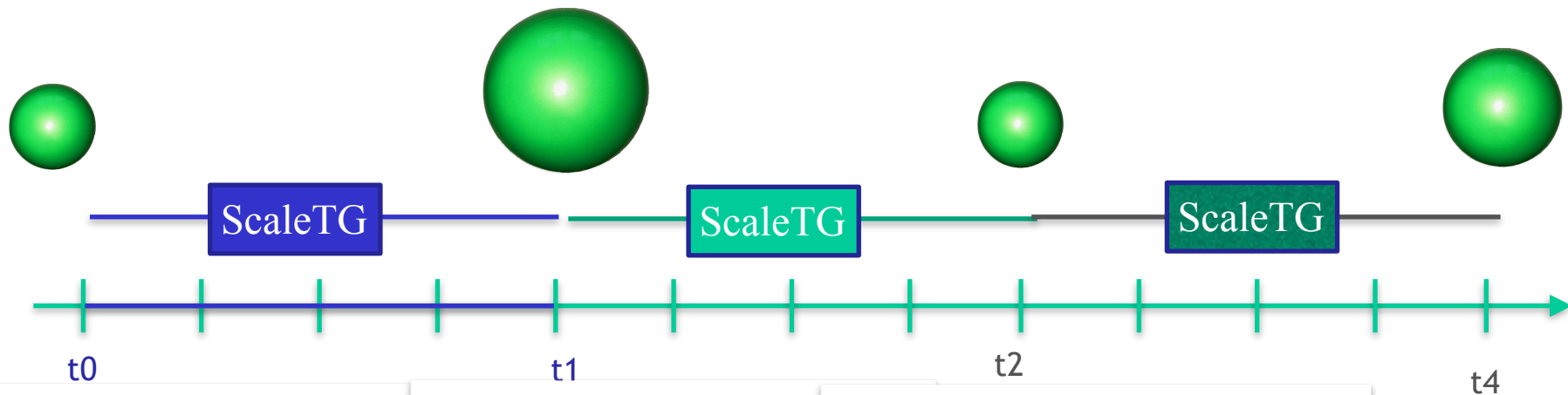
- n propietats (localització i gizmos)





# Visualization mapping (extensions de la pràctica)

- animacions de les dades (sèries temporals)



2.1621234	41.4056165	1000000
2.1677698	41.4034215	290000
2.135380995	41.389714424	307596
2.1564047	41.4062419	36500
2.185382	41.3855443	94830
2.178407934	41.381142750	307920
2.16018	41.40352	4214
2.175268003	41.379703771	450787
2.164072884	41.379806535	132024
2.1550105	41.3984803	51072
2.149045108	41.394592939	1300000
2.191080074	41.397724772	882703
2.169583037	41.375154356	725000
2.191159281	41.399440359	191125
2.175174697	41.380544047	1242221
2.175268003	41.379703771	818323

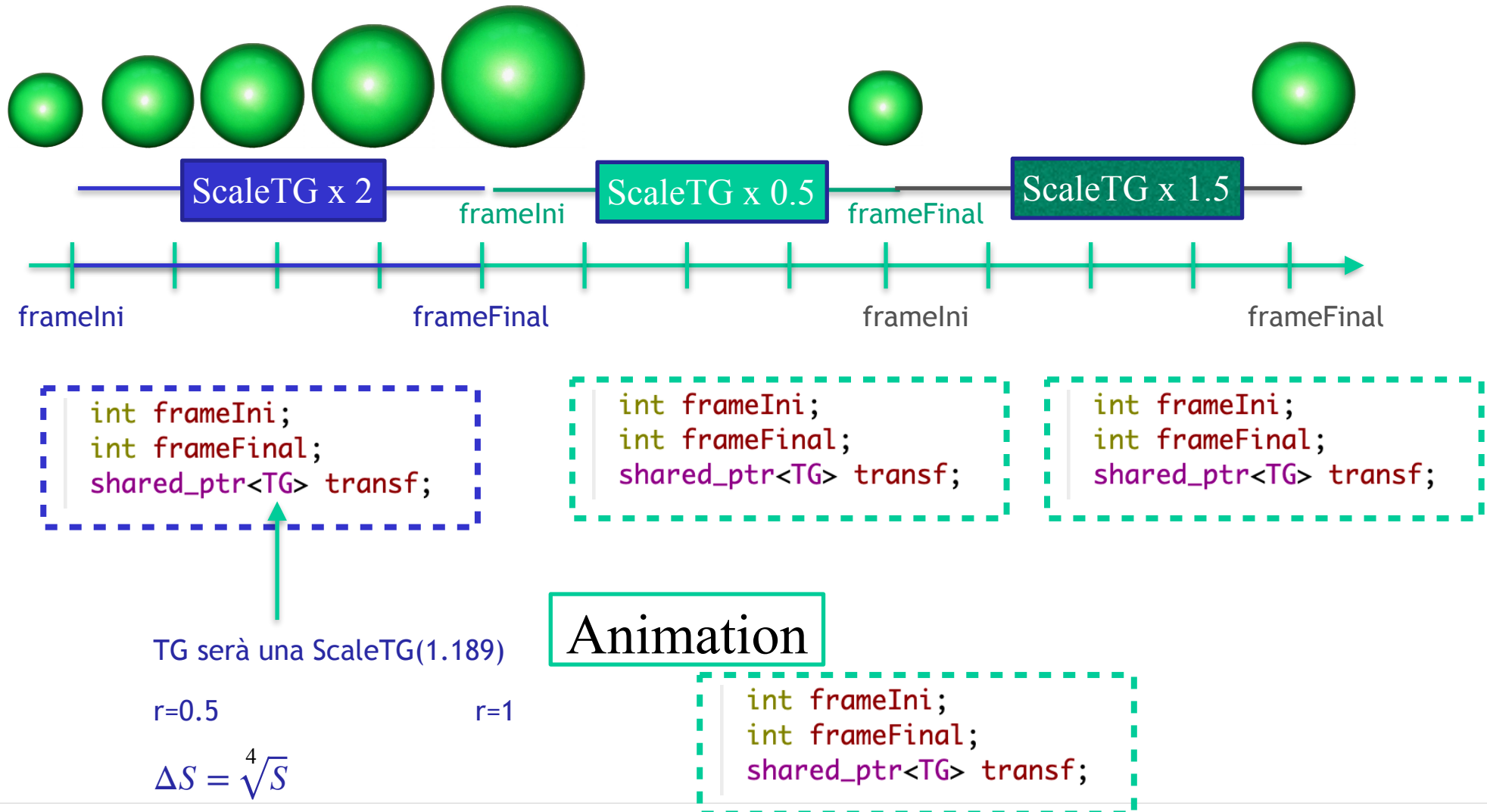
2.1621234	41.4056165	50003
2.1677698	41.4034215	190012
2.135380995	41.389714424	27
2.1564047	41.4062419	61650
2.185382	41.3855443	122483
2.178407934	41.381142750	50
2.16018	41.40352	232143
2.175268003	41.379703771	21
2.164072884	41.379806535	53
2.1550105	41.3984803	251724
2.149045108	41.394592939	12
2.191080074	41.397724772	32
2.169583037	41.375154356	73
2.191159281	41.399440359	92
2.175174697	41.380544047	22
2.175268003	41.379703771	11

2.1621234	41.4056165	81040
2.1677698	41.4034215	490200
2.135380995	41.389714424	475
2.1564047	41.4062419	316502
2.185382	41.3855443	294833
2.178407934	41.381142750	247
2.16018	41.40352	1121400
2.175268003	41.379703771	135
2.164072884	41.379806535	131
2.1550105	41.3984803	81724
2.149045108	41.394592939	140
2.191080074	41.397724772	112
2.169583037	41.375154356	890
2.191159281	41.399440359	119
2.175174697	41.380544047	122
2.175268003	41.379703771	898

2.1621234	41.4056165	101001
2.1677698	41.4034215	292001
2.135380995	41.389714424	137596
2.1564047	41.4062419	265012
2.185382	41.3855443	14833
2.178407934	41.381142750	179320
2.16018	41.40352	422142
2.175268003	41.379703771	1157387
2.164072884	41.379806535	202224
2.1550105	41.3984803	101072
2.149045108	41.394592939	1200000
2.191080074	41.397724772	1282703
2.169583037	41.375154356	954000
2.191159281	41.399440359	69235
2.175174697	41.380544047	132221
2.175268003	41.379703771	898313

# Visualization mapping (extensions de la pràctica)

- animacions de les dades (sèries temporals)



# Visualization mapping (extensions de la pràctica)

- animacions de les dades (sèries temporals)

## Run Animation

```
void MainWindow::runAnimation() {  
  
    int width, height;  
    vector<QImage> frames;  
  
    // Camera parameters from the Controller  
    auto camera = Controller::getInstance()->getSetUp()->getCamera();  
    width = camera->viewportX;  
    height = camera->viewportY;  
  
    // TODO: Canviar per incloure animacions diferents  
    // Ara es crea aqui una escena amb una esfera que té animacions,  
    // Caldria crear l'escena des de fitxer (opcionalment)  
    // es crida el render i es guarden  
    // tantes imatges com a frames s'han calculat (exemple amb MAXFRAMES = 5 a Animation.hh)  
  
    Controller::getInstance()->createScene(MAXFRAMES);  
    for (int i=0; i<MAXFRAMES; i++) {  
        QImage *image = new QImage(width, height, QImage::Format_RGB888);  
        Controller::getInstance()->update(i);  
        Controller::getInstance()->rendering(image);  
        frames.push_back(*image);  
    }  
  
    outputFile->saveAnimation(frames);  
    frames.clear();  
}
```

# Visualization mapping (extensions de la pràctica)

- animacions de les dades (sèries temporals)

## Run Animation

```
void MainWindow::runAnimation
```

```
    int width, height;  
    vector<QImage> frames;
```

```
    // Camera parameters from  
    auto camera = Controller:  
    width = camera->viewportX  
    height = camera->viewport
```

```
    // TODO: Canviar per incl  
    // Ara es crea aqui una e  
    // Caldria crear l'escena
```

```
    // es crida el render i es guarden  
    // tantes imatges com a frames s'ha calculat (exemple amb MAXFRAMES = 5 a Animation.hh)
```

```
    Controller::getInstance()->createScene(MAXFRAMES);
```

```
    for (int i=0; i<MAXFRAMES; i++) {  
        QImage *image = new QImage(width, height, QImage::Format_RGB888);  
        Controller::getInstance()->update(i);  
        Controller::getInstance()->rendering(image);  
        frames.push_back(*image);  
    }
```

```
    outputFile->saveAnimation(frames);  
    frames.clear();  
}
```

```
bool Controller::createScene(int nFrames) {
```

```
    //TODO DO Fase 3 opcional: Codi exemple amb animacions però que es pot canviar  
    // pel que creguis convenient
```

```
    auto sphere = make_shared<Sphere>(vec3(0, 0, -1), 0.5, 1.0);  
    sphere->setMaterial(make_shared<Lambertian>(vec3(0.5, 0.2, 0.7)));
```

```
    shared_ptr<Animation> anim = make_shared<Animation>();  
    anim->transf = make_shared<TranslateTG>(vec3(0.2));  
    sphere->addAnimation(anim);
```

```
    return true;
```



# Visualization mapping (extensions de la pràctica)

- animacions de les dades (sèries temporals)

## Run Animation

```
void MainWindow::runAnimation() {
```

```
    int width, height;  
    vector<QImage> frames;
```

```
    // Camera parameters from the Controller
```

```
    auto camera = Controller::getInstance()->getCamera();
```

```
    width = camera->viewportWidth();
```

```
    height = camera->viewportHeight();
```

```
    // TODO: Canviar per i
```

```
    // Ara es crea aqui un
```

```
    // Caldria crear l'esc
```

```
    // es crida el render
```

```
    // tantes imatges com
```

```
    Controller::getInstance()->createScene(MAX_FRAMES);
```

```
    for (int i=0; i<MAXFRAMES; i++) {
```

```
        QImage *image = new QImage(width, height, QImage::Format_RGB888);
```

```
        Controller::getInstance()->update(i);
```

```
        Controller::getInstance()->rendering(image);
```

```
        frames.push_back(*image);
```

```
    }
```

```
    outputFile->saveAnimation(frames);
```

```
    frames.clear();
```

```
}
```

## Scene

```
void Scene::update(int nframe) {
```

```
    for (unsigned int i = 0; i< objects.size(); i++) {
```

```
        objects[i]->update(nframe);
```

```
    }
```

```
}
```

# Visualization mapping (extensions de la pràctica)

- animacions de les dades (sèries temporals)

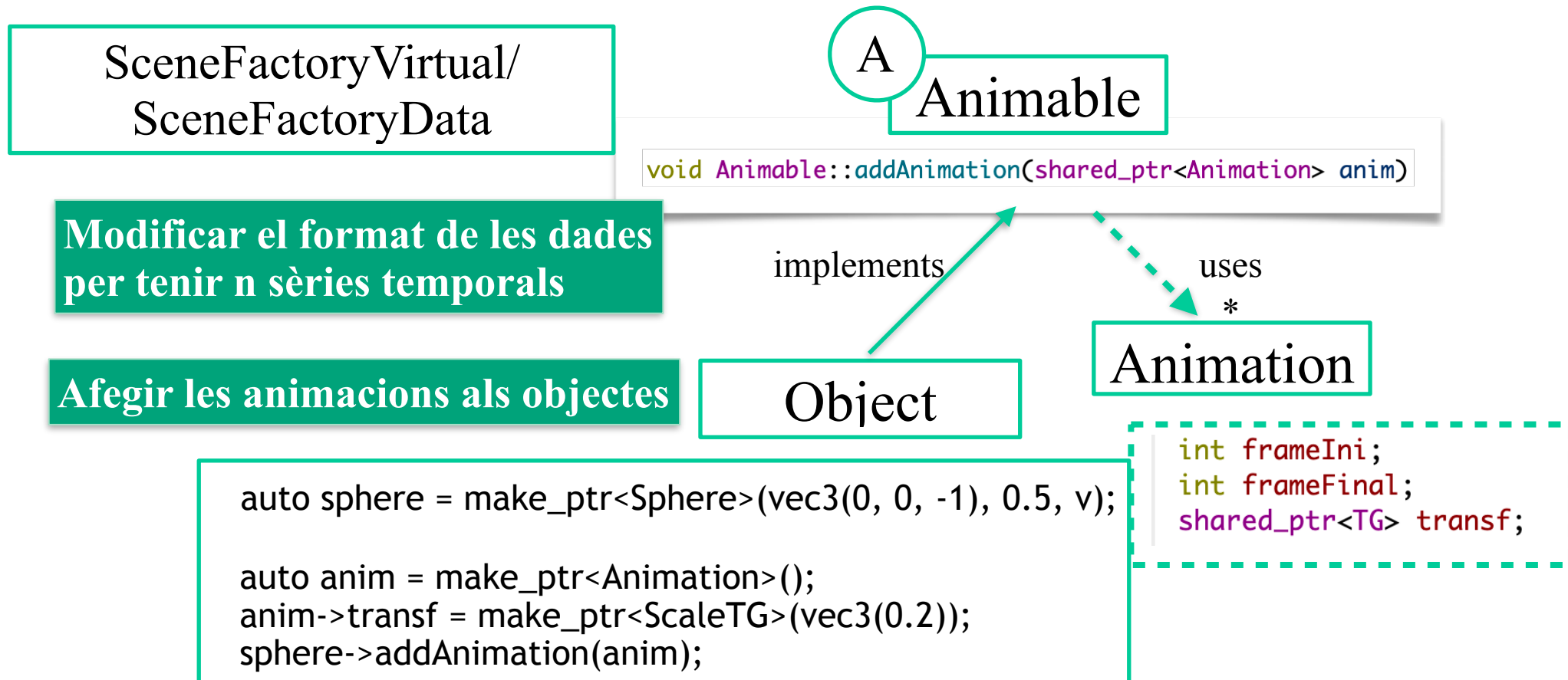
A  
Animable

```
void Animable::addAnimation(shared_ptr<Animation> anim) {  
    animFrames.push_back(anim);  
}
```

```
void Animable::update(int nframe) {  
  
    bool trobat = false;  
    int i;  
    for (i = 0; i<animFrames[animFrames.size()-1]->frameFinal && !trobat; i++)  
        trobat = animFrames[i]->frameFinal>=nframe;  
  
    aplicaTG(animFrames[i-1]->transf);  
}
```

# Visualization mapping (extensions de la pràctica)

- animacions de les dades (sèries temporals)



.json

TEMPORALVW o TEMPORALDATA

"numInstants": 10

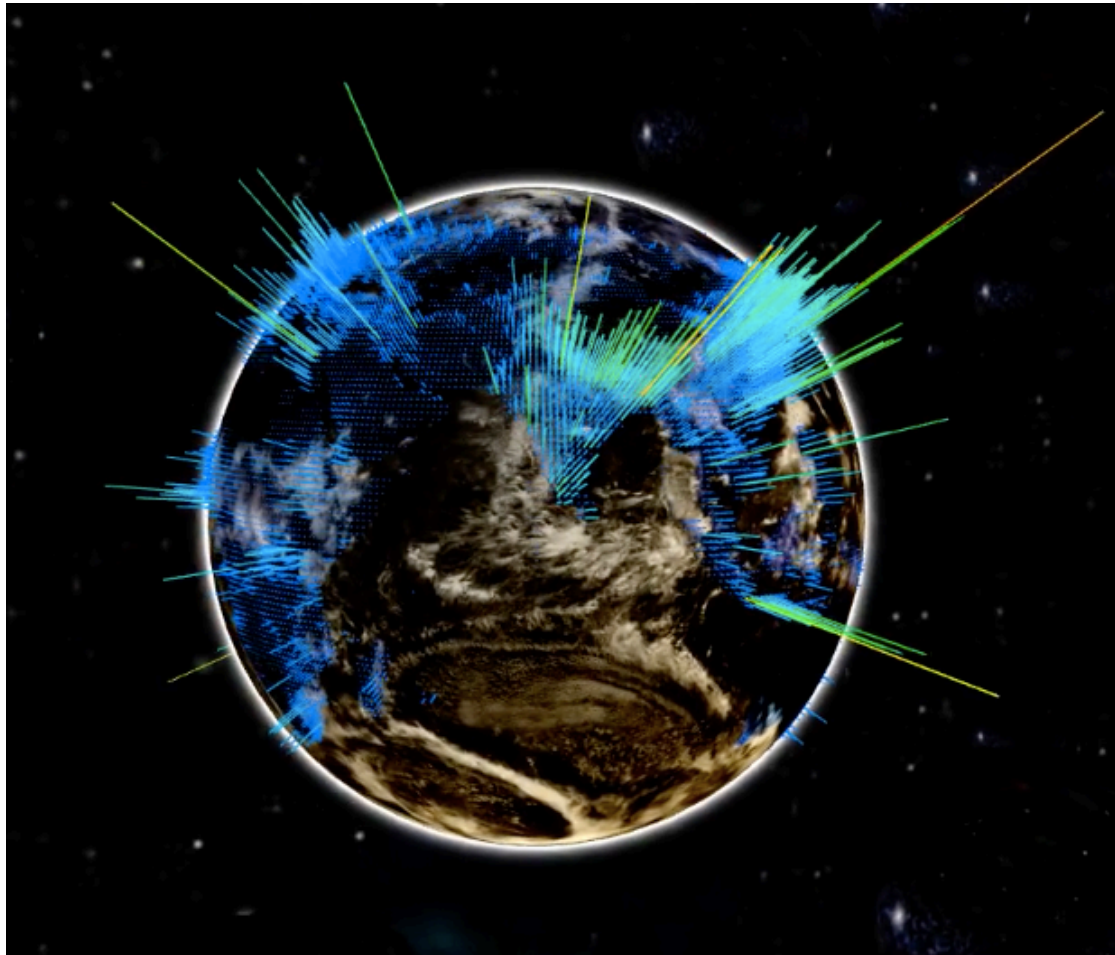
"translation": [ -1, -2, 3],

"scale": [2.0],

"rotation": [30, 1, 0, 0]

# Visualization mapping (extensions de la pràctica)

- Mapeig de les dades en una esfera



<https://www.covidvisualizer.com>