Simulated Photometry in CatSim

Users wishing to adapt CatSim for non-LSST projects will likely need to write getters that can calculate the magnitudes of objects in non-LSST bandpasses. This page explains how to do that. Those wishing to see an example implementation should look to

```
sims_catUtils/examples/exampleSDSScatalogs.py
```

which implements getters for SDSS bandpasses using CatSim.

Any implementation of photometry getters should occur within a mixin class that inherits from either PhotometryStars or PhotometryGalaxies. These are the mixins defined in

```
sims_catUtils/python/lsst/sims/catUtils/mixins/PhotometryMixin.py
```

that provide the basic infrastructure for calculating magnitudes in an arbitrary set of bandpasses. Specifically, they define the method _magnitudeGetter which takes a a set of arbitrary bandpasses (stored in an instance of the class BandpassDict) and returns the magnitudes of objects returned by CatSim in those bandpasses. We can see the use of this method demonstrated in the magnitudes getter from the example SDSS catalog above:

```
    @compound('sdss_u','sdss_g','sdss_r','sdss_i','sdss_z')
    def get_sdss_magnitudes(self):
        """
        An example getter for stellar magnitudes in SDSS bands
        """
        # Load a BandpassDict of SDSS bandpasses, if not done already
        if not hasattr(self, 'sdssBandpassDict'):
            bandpassNames = ['u','g','r','i','z']
            bandpassDir = os.path.join(getPackageDir('throughputs'),'sdss')
            bandpassRoot = 'sdss_'
            self.sdssBandpassDict = BandpassDict.loadTotalBandpassesFromFiles(bandpassNames,
                                                              bandpassRoot = bandpassRoot,
                                                              bandpassDir = bandpassDir)
        # Actually calculate the magnitudes
        return self._magnitudeGetter(self.sdssBandpassDict, self.get_sdss_magnitudes._colnames)
```

loadTotalBandpassesFromFiles

loadTotalBandpassesFromFiles() needs to be called the same way whether you are calculating the magnitudes for a star (which just has one component), or a galaxy (which will have separate magnitudes for the bulge, disk, and agn components). It is a class method of the class BandpassDict defined in

```
sims_photUtils/python/lsst/sims/photUtils/BandpassDict
```

Its arguments are

- bandpassNames – a list of the names of the bandpasses; these correspond to the keys for self.bandpassDict. See the example above for SDSS bands.
- bandpassDir – this is the absolute path to the directory in which the bandpass throughput files are kept.
- bandpassRoot – this is the root of the name of each bandpass throughput file

loadTotalBandpassesFromFiles() assumes that the bandpass throughput files are named like

```
for bp in bandpassNames:
    bandpassDir + bandpassRoot + bp + '.dat'
```

So

```
bandpassDir = '/my/bandpass/directory/'
bandpassRoot = 'sillyBandpass_'
bandpassNames = ['x', 'y', 'z']
```

corresponds to the bandpass throughput files

```
/my/bandpass/directory/sillyBandpass_x.dat
/my/bandpass/directory/sillyBandpass_y.dat
/my/bandpass/directory/sillyBandpass_z.dat
```

_magnitudeGetter()

The _magnitudeGetter() method differs depending on whether you are inheriting from PhotometryStars or PhotometryGalaxies. It takes as arguments

- componentName – for galaxies only. This specifies whether the getter is calculating magnitudes for the 'disk', 'bulge', or 'agn'
- bandpassDict – a BandpassDict instantiation containing the bandpasses to be integrated. The bandpasses in this dict must occur in the same order as the magnitudes are declared to the @compound getter.
- columnNameList – a list of the names of the columns to which these magnitudes correspond. This is so that the getter can associate the magnitudes with any variability model that may have been implemented. This list of magnitudes must occur in the same order as declared to the @compound getter.

The PhotometryStars case is very straightforward. In this case, the method reads in the SED file names, normalizations, and dust extinction parameters A (V) associated with each object, converts that data into an instantiation of the SedList class defined in

```
sims_photUtils/python/lsst/sims/photUtils/SedList.py
```

and integrates the SEDs in that SedList over the bandpasses in the BandpassDict passed to _magnitudeGetter(). Magnitudes are returned in a two dimensional numpy array in which each row corresponds to a different bandpass (in the order they occur in self.bandpassDict) and each column corresponds to a different object (i.e. a row of the database).

Note: because the _magnitudeGetter() returns magnitudes in the order specified by the input BandpassDict (which, in turn, is specified by the order of filters in the bandpassNames argument passed to loadTotalBandpassesFromFiles()), it is very important that the columns declared for the getter by @compound occur in the same order as the bandpasses in the list of bandpassNames passed to loadTotalBandpassesFromFiles().

In the case of galaxies, each component (bulge, disk, AGN) of the galaxy has its own SED and related parameters. This requires a suite of getter methods (including a getter method to combine the components into a total magntiude for the galaxy). In the case of our example SDSS catalog, this looks like

```
    @compound('sdss_bulge_u', 'sdss_bulge_g', 'sdss_bulge_r', 'sdss_bulge_i', 'sdss_bulge_z')
    def get_sdss_bulge_mags(self):
        """
        An example getter for SDSS bulge magnitudes
        """
        # load a BandpassDict of SDSS bandpasses, if not done already
        if not hasattr(self, 'sdssBandpassDict'):
            bandpassNames = ['u','g','r','i','z']
            bandpassDir = os.path.join(getPackageDir('throughputs'),'sdss')
            bandpassRoot = 'sdss_'
            self.sdssBandpassDict = BandpassDict.loadTotalBandpassesFromFiles(bandpassNames,
                                                              bandpassRoot = bandpassRoot,
                                                              bandpassDir = bandpassDir)
        # actually calculate the magnitudes
        return self._magnitudeGetter('bulge', self.sdssBandpassDict,
                                     self.get_sdss_bulge_mags._colnames)

    @compound('sdss_disk_u', 'sdss_disk_g', 'sdss_disk_r', 'sdss_disk_i', 'sdss_disk_z')
    def get_sdss_disk_mags(self):
        """
        An example getter for SDSS disk magnitudes
        """
        # load a BandpassDict of SDSS bandpasses, if not done already
        if not hasattr(self, 'sdssBandpassDict'):
            bandpassNames = ['u','g','r','i','z']
            bandpassDir = os.path.join(getPackageDir('throughputs'),'sdss')
            bandpassRoot = 'sdss_'
            self.sdssBandpassDict = BandpassDict.loadTotalBandpassesFromFiles(bandpassNames,
                                                              bandpassRoot = bandpassRoot,
                                                              bandpassDir = bandpassDir)
        # actually calculate the magnitudes
        return self._magnitudeGetter('disk', self.sdssBandpassDict,
                                     self.get_sdss_disk_mags._colnames)

    @compound('sdss_agn_u', 'sdss_agn_g', 'sdss_agn_r', 'sdss_agn_i', 'sdss_agn_z')
    def get_sdss_agn_mags(self):
        """
```

```
        An example getter for SDSS AGN magnitudes
        """
        # load a BandpassDict of SDSS bandpasses, if not done already
        if not hasattr(self, 'sdssBandpassDict'):
            bandpassNames = ['u','g','r','i','z']
            bandpassDir = os.path.join(getPackageDir('throughputs'),'sdss')
            bandpassRoot = 'sdss_'
            self.sdssBandpassDict = BandpassDict.loadTotalBandpassesFromFiles(bandpassNames,
                                                              bandpassRoot = bandpassRoot,
                                                              bandpassDir = bandpassDir)
        # actually calculate the magnitudes
        return self._magnitudeGetter('agn', self.sdssBandpassDict,
                                     self.get_sdss_agn_mags._colnames)

    @compound('sdss_u', 'sdss_g', 'sdss_r', 'sdss_i', 'sdss_z', 'sdss_y')
    def get_sdss_total_mags(self):
        """
        An example getter for total galaxy magnitudes in SDSS bands
        """
        idList = self.column_by_name('uniqueId')
        numObj = len(idList)
        output = []
        # Go through each of the calculated columns.  Find the bulge, disk, and agn
        # magnitudes corresponding to each bandpass.  Sum them using the
        # sum_magnitudes method
        for columnName in self.get_sdss_total_mags._colnames:
            if columnName not in self._actually_calculated_columns:
                sub_list = [numpy.NaN]*numObj
            else:
                bandpass = columnName[-1]
                bulge = self.column_by_name('sdss_bulge_%s' % bandpass)
                disk = self.column_by_name('sdss_disk_%s' % bandpass)
                agn = self.column_by_name('sdss_agn_%s' % bandpass)
                sub_list = self.sum_magnitudes(bulge=bulge, disk=disk, agn=agn)
            output.append(sub_list)
        return numpy.array(output)
```

The _magnitudeGetter for galaxies behaves exactly like the _magnitudeGetter for stars, except that it takes the extra argument componentName ('bulge', 'disk', or 'agn') so that it knows which list of SEDs to load and integrate over the BandpassDict.

Return to the main Catalog Simulations documentation page.